Quantum Lambda Calculi with Classical Control: Syntax and Expressive Power

Ugo Dal Lago
Dipartimento di Scienze dell'Informazione
Università di Bologna
dallago@cs.unibo.it

Andrea Masini
Dipartimento di Informatica
Università di Verona
andrea.masini@univr.it

Margherita Zorzi
Dipartimento di Informatica
Università di Verona
zorzim@sci.univr.it

Abstract

We study an untyped λ–calculus with quantum data and classical control. This work stems from previous proposals by Selinger and Valiron and by Van Tonder. We focus on syntax and expressiveness, rather than (denotational) semantics. We prove subject reduction, confluence and a standardization theorem. Moreover, we prove the computational equivalence of the proposed calculus with a suitable class of quantum circuit families.

1 Introduction

Quantum computing was conceived at the beginning of the eighties, starting from an idea by Feynman [6]. It defines an alternative computational paradigm, based on quantum mechanics [2] rather than digital electronics. The first proposal for a quantum abstract computer is due to Deutsch, who introduced quantum Turing machines [5]. Other quantum computational models have been subsequently defined by Yao (quantum circuits, [19]) and Knill (quantum random access machines, [8]).

The introduction of quantum abstract machines made it possible to develop a complexity theory of quantum computation. The most remarkable result in quantum complexity theory has been obtained by Shor, who showed that integers can be factorized in polynomial time [15]. Shor's algorithm, like the majority of quantum algorithmics, is defined as a quantum circuit family generated by a classical device.

Nowadays, what are the main challenges in quantum computing? A lot of research is being devoted to understanding whether quantum computation can provide efficient algorithms for classically intractable problems. In the last years, the impressive results obtained in this area (e.g. Shor's fast factoring algorithm) have stimulated the development of quantum programming languages. The situation is not as easy as in the classical case. In addition to the concrete technical problems (up to now it is difficult to build even very simple quantum circuits) there is the necessity of developing adequate theoretical bases for quantum programming languages — even with the best will in the world it is hard to look at quantum Turing machines as a basis for programming. *This paper is an attempt to give a contribution to the definition of a (higher-order) quantum computational model.*

The first attempt to define a quantum functional programming language has been done (to our knowledge) in two unpublished papers by Maymin [9, 10]. Selinger [13] rigorously defined a first-order quantum functional language. Another interesting proposal in the framework of first-order quantum functional languages is the language QML of Altenkirch and Grattage [1].

Focusing on higher-order functional programming languages, at least two distinct proposals have already appeared in the literature: that by Selinger and Valiron [14] and the one by Van Tonder [17]. These two approaches seems to go in orthogonal directions: in the language proposed by Selinger and Valiron data (registers of qubits) are superimposed while control (lambda terms) is classical, whereas the approach of Van Tonder is based on the idea of putting *arbitrary λ–terms in superposition*. But, *is this the right picture*? In order to give an answer let us examine more closely the two approaches.

Selinger and Valiron's Approach. The main goal of the work of Selinger and Valiron is to give the basis of a typed quantum functional language (with types in propositional multiplicative and exponential linear logic). The great merit of

Selinger and Valiron is to have defined a language where only data are superposed, and where programs live in a standard classical world. In particular, it is not necessary to have "exotic" objects such as λ–terms in superposition. The approach is well condensed by the slogan: *"classical control + quantum data"*. The proposed calculus, here dubbed λ_{sv}, is based on a call-by-value λ–calculus enriched with constants for unitary transformations and an explicit measurement operator.

Unfortunately, the expressive power of λ_{sv} has not been studied yet. The crucial issue is the following: can we compare the expressive power of λ_{sv} with the one of any well known computational model (e.g. quantum Turing machines or quantum circuits families)?

Van Tonder's Approach. The calculus introduced by Van Tonder [17], called λ_q, has the same motivation and a number of *immediate similarities* with λ_{sv}, noticeably, the exploitation of linear types in controlling both copying and erasing of terms.

But there is a glaring difference between λ_q and λ_{sv}. In fact it seems that λ_q *allows by design arbitrary superpositions of λ-terms.* In our opinion the essence of the approach of Van Tonder is in lemma 5.1 of [17], where it is stated that "*two terms t_1, t_2 in superposition differ only for qubits values*". Moreover, if t_1 reduces to t_1' and t_2 reduces to t_2', the reduced redex in t_1 is (up to quantum bits) the same redex reduced in t_2. This means λ_q has classical control, too: it is not possible to superimpose terms differing in a remarkable way, i.e. terms with a different computational evolution.

The weak point of Van Tonder's paper, is that some results and proofs are given too informally. In particular, the paper argues that the proposed calculus is computationally equivalent to quantum Turing machines without giving a detailed proof and, more importantly, without specifying which class of quantum Turing machines is considered (this is not pedantry, since there isn't anything like a Church-Turing thesis in quantum computation [12]). But clearly, such a criticism does not invalidate the foundational importance of the approach.

Our Proposal. Our goal is to propose an alternative quantum computational paradigm, proving its computational equivalence with quantum circuits families.

Our work can be seen both as a *continuation* and *extension* of the two proposals we have just described.

- It is a *continuation* because we propose a quantum λ–calculus with classical control and quantum data. We use a syntax for terms and configurations inspired by that of Selinger and Valiron and moreover we implicitly use linear logic in a way similar to Van Tonder's λ_q.
- It is an *extension* because we have focused on the syntactical study of the calculus. Important classical properties such as *subject reduction* and *confluence* are given. Moreover a novel *quantum standardization theorem* is given. The *expressive power* of the calculus has been studied in a detailed way (to our knowledge, it is the first time such a study has been done for a quantum λ–calculus). In order to face the expressive power problem, we prove the *equivalence between our calculus and quantum circuit families*.

We have chosen λ–calculus as a basis of our proposal for a number of reasons:

- first of all, quantum computability and complexity theory are quite underdeveloped compared to their classical counterparts; in particular, there is almost no result relating classes of (first-order) functions definable in pure and typed λ–calculi with classes of functions from computability and complexity theory (in contrast with classical computability theory [7]);
- we hope that our proposal will contribute to the development of a "quantum computationally complete" functional programming language. Quantum Turing machines and quantum circuit families are good for computability theory, but quite useless from a programming perspective;
- we believe that the higher–order nature of λ–calculi could be useful for understanding the interactions between the classical world (the world of terms) and the quantum world (quantum registers).

The paper is structured as follows:

- in Section 2 we give the mathematical background on Hilbert Spaces (in order to define quantum registers);
- in Section 3, a λ–calculus, called the Q-calculus, is introduced. The Q-calculus has classical control and quantum data. The calculus is untyped, but is equipped with well-formation judgments for terms based on the formulation of linear logic as proposed in [18];
- in Section 4 we syntactically study the Q-calculus by means of a suitable formulation of subject reduction theorem and confluence theorems. Noticeably, a configuration is strongly normalizing iff it is weakly normalizing;
- in section 5 a further result on the dynamics of the Q-calculus is given: for each terminating computation there is another "canonical", equivalent computation where computational steps are performed in the following order:
  1. first, classical reductions: in this phase the quantum register is empty and all the computations steps are classical;
  2. secondly, reductions that build the quantum register;
  3. and finally quantum reductions, applying unitary transformations to the quantum register.

Such a property is formally ensured by means of a suitable standardization theorem and sheds some further light on the dynamics of computation;

- in Sections 6 we study in detail the equivalence of the Q-calculus with Quantum Circuit Families. The equivalence proofs are based on the standardization theorem and on suitable encodings.

2 Mathematical Structures

This section is devoted to mathematical preliminaries. Clearly, we cannot hope to be completely self-contained here. See [11] for an excellent introduction to quantum computing.

2.1 Quantum Computing Basics

We informally recall here the basic notations on qubits and quantum registers (see [11] for a detailed introduction). In the next subsection such notations will be (re)defined in a rigorous way.
The basic unit of quantum computation is called *quantum bit*, or *qubit* for short . The more direct way to represent a quantum bit is by an unitary vector in the 2-dimensional Hilbert space $\mathbb{C}^2$. Let us denote with $|0\rangle$ and $|1\rangle$ the elements of an orthonormal basis of $\mathbb{C}^2$.

The states $|0\rangle$ and $|1\rangle$ of a qubit can be seen as the correspondent states of a classical bit. A qubit, however, can be in other states, different from $|0\rangle$ and $|1\rangle$. In fact, every linear combination $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$, and $|\alpha|^2 + |\beta|^2 = 1$, can be a possible qubit state. These states are *superpositions*, and the two values α and β are called *amplitudes*.

While we can determine the state of a classical bit, for a qubit we can't establish with the same precision what is it's quantum state, namely the values of α and β: quantum mechanics says that a measurement of a qubit with state $\alpha|0\rangle + \beta|1\rangle$ has the effect of changing the state of the qubit to $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$.

In computational models, we need a generalization of the notion of a qubit, namely the so called *quantum register* [12, 13, 14, 17]. A quantum register of arity *n* is a normalized vector in $\otimes_{i=1}^{n}\mathbb{C}^2$. We fix an orthonormal basis of $\otimes_{i=1}^{n}\mathbb{C}^2$, namely $\{|i\rangle | i \text{ is a binary string of length } n\}$. For example $1/\sqrt{2}|01\rangle + 1/\sqrt{2}|00\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ is a quantum register of two qubits.

An important property of quantum registers of n qubits is the fact that it is not always possible to decompose it into n isolated qubits (mathematically, this means that we are no able to describe the global state as the tensor product of the single states). These particular states are called *entangled* and enjoy properties that we can't find in any object of classical physics. If (the state of) n qubits are entangled, they behave as connected, independently from the real physical distance. The strength of quantum computation is essentially based on the existence of entangled states.

2.2 Hilbert Spaces and Quantum Registers

Even if Hilbert spaces of the shape $\otimes_{i=1}^{n}\mathbb{C}^2 (\simeq \mathbb{C}^{2^n})$ are commonly used when defining quantum registers, other spaces will be defined here. As we will see, they allow to handle very naturally the interaction between variable names in λ–terms and superimposed data.

A *quantum variable set (qvs)* is a finite set of quantum variables (ranged over by variables like p, r and q).

Definition 1 (Hilbert Spaces on $\mathcal{V}$). Let $\mathcal{V}$ a *qvs* (possibly empty) of cardinality $\#\mathcal{V} = n$, with $\mathcal{H}(\mathcal{V}) = \{\phi |\ \phi : \{0,1\}^{\mathcal{V}} \to \mathbb{C}\}$ we will denote the Hilbert Space of dimension 2^n equipped with:

i. An *inner sum* $+ : \mathcal{H}(\mathcal{V}) \times \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ defined by $(\phi + \psi)(f) = \phi(f) + \psi(f)$;
ii. A *multiplication by a scalar* $\cdot : \mathbb{C} \times \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$
defined by $(c \cdot \phi)(f) = c \cdot (\phi(f))$;
iii. An *inner product* $\langle\ ,\ \rangle : \mathcal{H}(\mathcal{V}) \times \mathcal{H}(\mathcal{V}) \to \mathbb{C}$
defined by $\langle \phi, \psi \rangle = \sum_{f \in \{0,1\}^{\mathcal{V}}} \phi(f)^* \psi(f)$.

The space is equipped with the *orthonormal basis* $\mathcal{B}(\mathcal{V}) = \{|f\rangle : f \in \{0,1\}^{\mathcal{V}}\}$.[1] We call *standard* such a basis. For example, the standard basis of the space $\mathcal{H}(\{p,q\})$ is $\{|p \mapsto 0, q \mapsto 0\rangle, |p \mapsto 0, q \mapsto 1\rangle, |p \mapsto 1, q \mapsto 0\rangle, |p \mapsto 1, q \mapsto 1\rangle\}$.

Let $\mathcal{V}' \cap \mathcal{V}'' = \emptyset$. With $\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'')$ we denote the tensor product (defined in the usual way) of $\mathcal{H}(\mathcal{V}')$ and $\mathcal{H}(\mathcal{V}'')$. If $\mathcal{B}(\mathcal{V}') = \{|f_i\rangle : i \leq 2^{n-1}\}$ and $\mathcal{B}(\mathcal{V}'') = \{|g_j\rangle : j \leq 2^{m-1}\}$ are the orthonormal bases respectively of $\mathcal{H}(\mathcal{V}')$ and $\mathcal{H}(\mathcal{V}'')$

[1] $|f\rangle : \{0,1\}^{\mathcal{V}} \to \mathbb{C}$ is defined by: $|f\rangle(g) = \begin{cases} 1 \text{ if } f = g \\ 0 \text{ if } f \neq g \end{cases}$

then $\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'')$ is equipped with the orthonormal basis $\{|f_i\rangle \otimes |g_j\rangle : i \leq 2^{n-1}, j \leq 2^{m-1}\}$. We will abbreviate $|f\rangle \otimes |g\rangle$ with $|f, g\rangle$.

It is easy to show that if $\mathcal{V}' \cap \mathcal{V}'' = \emptyset$ then there is a standard *isomorphism*

$$\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'') \stackrel{i_s}{\simeq} \mathcal{H}(\mathcal{V}' \cup \mathcal{V}'').$$

In the rest of the paper we will assume to work up-to such an isomorphism[2].

As for the case of $\mathbb{C}^{2^n}$, we need to define the notion of a *quantum register*.

Definition 2 (Quantum Register)**.** Let $\mathcal{V}$ be a qvs, a *quantum register* is a normalized vector in $\mathcal{H}(\mathcal{V})$.

In particular if $\mathcal{Q}' \in \mathcal{H}(\mathcal{V}')$ and $\mathcal{Q}'' \in \mathcal{H}(\mathcal{V}'')$ are two quantum registers, with a little abuse of language (authorized by the previous stated isomorphism) we will say that $\mathcal{Q}' \otimes \mathcal{Q}''$ is a quantum register in $\mathcal{H}(\mathcal{V}' \cup \mathcal{V}'')$.

Quantum computing is essentially based on the application of unitary operators to quantum registers. A linear operator $U : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ is called *unitary* if for all $\phi, \psi \in \mathcal{H}(\mathcal{V})$, $\langle U(\phi), U(\psi)\rangle = \langle \phi, \psi\rangle$ The tensor product of unitary operators is defined as follows: $(U \otimes V)(\phi \otimes \psi) = U(\phi) \otimes U(\psi)$.

Since we are interested in effective computability, we must restrict the class of admissible unitary transforms. Following Bernstein and Vazirani [3] let us define the set $\mathbf{P}\mathbb{C}$ of poly–time computable complex numbers:

Definition 3. A real number $x \in \mathbb{R}$ is *polynomial-time computable* (in $\mathbf{P}\mathbb{R}$) iff there is a deterministic Turing machine which on input 1^n computes a binary representation of an integer $m \in \mathbb{Z}$ such that $|m/2^n - x| \leq 1/2^n$. A complex number $z = x + iy$ is *polynomial-time computable* (in $\mathbf{P}\mathbb{C}$) iff $x, y \in \mathbf{P}\mathbb{R}$.

Let $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ be an unitary operator. U is called computable if $U((\mathbf{P}\mathbb{C})^{2^n}) \subseteq (\mathbf{P}\mathbb{C})^{2^n}$. Let $\mathcal{U}$ be the set of all computable operators; it is immediate to observe that $\mathcal{U}$ is effectively enumerable. In the rest of the paper we assume to work with a fixed effective enumeration $(\mathbf{U}_i)_{i<\omega}$ of $\mathcal{U}$.

Definition 4. A quantum register in $\phi \in \mathcal{H}(\mathcal{V})$ is *computable* if $\phi : \{0, 1\} \to \mathbf{P}\mathbb{C}$. A unitary operator $U : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ is called "computable" if for each computable quantum register ϕ, $U(\phi)$ is computable.

Let $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ be a computable operator and let $\langle q_0, \ldots, q_{n-1}\rangle$ be a sequences of distinguished variables. U and $\langle q_0, \ldots, q_{n-1}\rangle$ induce a computable operator $U_{\langle q_0,\ldots,q_{n-1}\rangle} : \mathcal{H}(\{q_0, \ldots, q_{n-1}\}) \to \mathcal{H}(\{q_0, \ldots, q_{n-1}\})$ defined as follows: if $|f\rangle = |q_{j_0} \mapsto b_{j_0}, \ldots, q_{j_{n-1}} \mapsto b_{j_{n-1}}\rangle$ is an element of the orthonormal basis of $\mathcal{H}(\{q_0, \ldots, q_{n-1}\})$, then

$$U_{\langle q_0,\ldots,q_{n-1}\rangle}|f\rangle \stackrel{def}{=} U|b_{j_0}, \ldots, b_{j_{n-1}}\rangle.$$

Let $\mathcal{V}' = \{q_{i_0}, \ldots, q_{i_k}\} \subseteq \mathcal{V}$. We naturally extend (by suitable standard isomorphisms) the unitary operator $U_{\langle q_{j_0},\ldots,q_{j_k}\rangle} : \mathcal{H}(\mathcal{V}') \to \mathcal{H}(\mathcal{V}')$ to the unitary operator $U_{\langle\langle q_{j_0},\ldots,q_{j_k}\rangle\rangle} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ that acts as the identity on variables not in $\mathcal{V}'$ and as $U_{\langle q_{j_0},\ldots,q_{j_k}\rangle}$ on variables in $\mathcal{V}'$.

Example 1. *Let us consider the the standard computable operator* $\mathbf{cnot} : \mathbb{C}^2 \otimes \mathbb{C}^2 \to \mathbb{C}^2 \otimes \mathbb{C}^2$. *Intuitively, the* ***cnot*** *operator complements the target bit (the second one) if the control bit is 1, otherwise does not perform any action:*

$$\begin{array}{lcllcl} \mathbf{cnot}|00\rangle & = & |00\rangle & \mathbf{cnot}|10\rangle & = & |11\rangle \\ \mathbf{cnot}|01\rangle & = & |01\rangle & \mathbf{cnot}|11\rangle & = & |10\rangle \end{array}$$

Let us fix the sequence $\langle p, q\rangle$ *of variables,* $\mathbf{cnot}$ *induces the operator* $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle} : \mathcal{H}(\{p, q\}) \to \mathcal{H}(\{p, q\})$ *such that:*

$$\begin{array}{rcl} \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 0, p \mapsto 0\rangle & = & |q \mapsto 0, p \mapsto 0\rangle; \\ \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 0, p \mapsto 1\rangle & = & |q \mapsto 1, p \mapsto 1\rangle; \\ \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 1, p \mapsto 0\rangle & = & |q \mapsto 1, p \mapsto 0\rangle; \\ \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 1, p \mapsto 1\rangle & = & |q \mapsto 0, p \mapsto 1\rangle. \end{array}$$

Please note that $|q \mapsto c_1, p \mapsto c_2\rangle = |p \mapsto c_2, q \mapsto c_1\rangle$ *(consequently* $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto c_1, p \mapsto c_2\rangle = \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|p \mapsto c_2, q \mapsto c_1\rangle$*). On the other hand, the operators* $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle}$ *and* $\mathbf{cnot}_{\langle\langle q,p\rangle\rangle}$ *are different: both act as controlled not, but* $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle}$ *uses* p *as control bit while* $\mathbf{cnot}_{\langle\langle q,p\rangle\rangle}$ *uses* q*.*

[2] in particular, if $\mathcal{Q} \in \mathcal{H}(\mathcal{V})$, $r \notin \mathcal{V}$ and $|r \mapsto c\rangle \in \mathcal{H}(\{r\})$ then $\mathcal{Q} \otimes |r \mapsto c\rangle$ will denote the element $i_s(\mathcal{Q} \otimes |r \mapsto c\rangle) \in \mathcal{H}(\mathcal{V} \cup \{r\})$

3 The Q-calculus

Let us associate to each computable unitary operator $\mathbf{U_i} \in \mathcal{U}$ a symbol U_i

Terms

The set of the *term expressions*, or *terms* for short, is defined by the following grammar:

$$\begin{array}{lcll} x & ::= & v_0, v_1, \ldots & \text{classical variables} \\ r & ::= & r_0, r_1, \ldots & \text{quantum variables} \\ \pi & ::= & x \mid \langle x_1, \ldots, x_n \rangle & \text{patterns} \\ B & ::= & 0 \mid 1 & \text{boolean constants} \\ U & ::= & U_0, U_1, \ldots & \text{unitary operators} \\ C & ::= & B \mid U & \text{constants} \\ M & ::= & x \mid r \mid !(M) \mid C \mid \mathtt{new}(M) \mid (M_1)M_2 \mid & \\ & & \langle M_1, \ldots, M_n \rangle \mid \lambda !x.M \mid \lambda \pi.M & \\ & & & \text{terms (where } n \geq 2) \end{array}$$

We assume to work modulo variable renaming, i.e., terms are equivalence classes modulo α-conversion. Substitution up to α-equivalence is defined in the usual way. Let us denote with $\mathbf{Q}(M_1, \ldots, M_k)$ the set of quantum variables occurring in $M_1, \ldots, M_k$. Notice that:

- Variables are either *classical* or *quantum*: the first ones are the usual variables of lambda calculus, while each quantum variable refers to a qubit in the underlying quantum register (to be defined shortly).
- There are two sorts of constants as well, namely *boolean constants* (0 and 1) and *unitary operators*: the first ones are useful for generating qubits and play no role in classical computations, while unitary operators are applied to (tuples of) quantum variables when performing quantum computation.
- The term constructor $\mathtt{new}(\cdot)$ creates a new qubit when applied to a boolean constant.

The rest of the calculus is a standard linear lambda calculus, similar to the one introduced in [18]. Patterns (and, consequently, lambda abstractions) can only refer to classical variables.

There is not any measurement operator in the language. We will comment on that in Section 7.

3.1 Judgements and Well–Formed Terms

An *environment* Γ is a (possibly empty) multiset $\Pi \cup \Delta \cup \Theta$ where Π is a (possibly empty) multiset $\pi_1, \ldots, \pi_n$ of patterns and Δ is a (possibly empty) multiset $!x_1, \ldots, !x_n$ (where each x_i is a classical variable), and Θ is a (possibly empty) multiset of quantum variables. We require that each variable name occurs at most once in Γ. With $!\Gamma$ we denote the environment $!x_1, \ldots, !x_n$ whenever Γ is $x_1, \ldots, x_n$.

A *judgment* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term. We say that a judgement $\Gamma \vdash M$ is *well*

$$\frac{}{\vdash C}\, const \qquad \frac{}{r \vdash r}\, qp-var \qquad \frac{}{x \vdash x}\, classic-var \qquad \frac{\Gamma \vdash M}{\Gamma, !x \vdash M}\, weak \qquad \frac{\Gamma, !x, !y \vdash M}{\Gamma, !z \vdash M\{z/x, z/y\}}\, contr$$

$$\frac{!\Gamma \vdash M}{!\Gamma \vdash !M}\, prom \qquad \frac{\Gamma, x \vdash M}{\Gamma, !x \vdash M}\, der \qquad \frac{\Gamma, x_1, \ldots, x_k \vdash M}{\Gamma, \langle x_1, \ldots, x_k \rangle \vdash M}\, Ltens \qquad \frac{\Gamma_1 \vdash M_1 \cdots \Gamma_k \vdash M_k}{\Gamma_1, \ldots, \Gamma_k \vdash \langle M_1, \ldots, M_k \rangle}\, Rtens$$

$$\frac{\Gamma \vdash M}{\Gamma \vdash \mathtt{new}(M)}\, \mathtt{new} \qquad \frac{\Gamma, \pi \vdash M}{\Gamma \vdash \lambda \pi.M} \multimap I \qquad \frac{\Gamma, !x \vdash M}{\Gamma \vdash \lambda !x.M} \to I \qquad \frac{\Gamma_1 \vdash M_1 \quad \Gamma_2 \vdash M_2}{\Gamma_1, \Gamma_2 \vdash (M_1)M_2}\, app$$

Figure 1. Well Forming Rules

formed (*notation:* $\triangleright\, \Gamma \vdash M$) if it is derivable by means of the *well forming rules* in Figure 1; with $d \triangleright\, \Gamma \vdash M$ we denote that d is a derivation of the well formed judgement $\Gamma \vdash M$. If $\Gamma \vdash M$ is *well formed* we say also that the term M is well formed with respect to the environment Γ. We say that a term M is *well formed* if the judgment $\mathbf{Q}(M) \vdash M$ is well formed.

Proposition 1. *If a term M is well formed then all the classical variables in it are bounded.*

4 Computations

A *preconfiguration* is a triple $[\mathcal{Q}, \mathcal{QV}, M]$ where:

- $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$;
- $\mathcal{QV}$ is a finite quantum variable set such that $\mathbf{Q}(M) \subseteq \mathcal{QV}$;
- M is a term.

Let $\theta : \mathcal{QV} \to \mathcal{QV}'$ be a function from a set of quantum variables $\mathcal{QV}$ to another set of quantum variables $\mathcal{QV}'$. Then we can extend θ to any term whose quantum variables are included in $\mathcal{QV}$: $\theta(M)$ will be identical to M, except on quantum variables, which are changed according to θ itself. Observe that $\mathbf{Q}(\theta(M)) \subseteq \mathcal{QV}'$. Similarly, θ can be extended to a function from $\mathcal{H}(\mathcal{QV})$ to $\mathcal{H}(\mathcal{QV}')$ in the obvious way.

Definition 5. Two preconfigurations $[\mathcal{Q}, \mathcal{QV}, M]$ and $[\mathcal{Q}', \mathcal{QV}', M']$ are equivalent iff there is a bijection $\theta : \mathcal{QV} \to \mathcal{QV}'$ such that $\mathcal{Q}' = \theta(\mathcal{Q})$ and $M' = \theta(M)$. If a preconfiguration C is equivalent to C', then we will write $C \equiv C'$. The relation $\equiv$ is an equivalence relation.

A *configuration* is an equivalence class of preconfigurations modulo the relation $\equiv$. Let $\mathcal{C}$ be the set of configurations.

Remark 1. The way configurations have been defined, namely quotienting preconfigurations over $\equiv$, is very reminiscent of usual α-conversion in lambda-terms.

Let $\mathscr{L} = \{\mathsf{Uq}, \mathsf{new}, \mathsf{l}.\beta, \mathsf{q}.\beta, \mathsf{c}.\beta, \mathsf{l.cm}, \mathsf{r.cm}, \mathsf{t}_i\}$. The set $\mathscr{L}$ will be ranged over by α, β, γ. For each $\alpha \in \mathscr{L}$, we can define a reduction relation $\to_\alpha \subseteq \mathcal{C} \times \mathcal{C}$ by means of the rules in Figure 2. For any subset $\mathscr{S}$ of $\mathscr{L}$, we can construct a relation

$$\frac{[\mathcal{Q}, \mathcal{QV}, M_i] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M_i'] \quad [\mathcal{Q}, \mathcal{QV}, \langle M_1, \ldots, M_i, \ldots, M_k\rangle] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, , \langle M_1, \ldots, M_i, \ldots, M_k\rangle] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \langle M_1, \ldots, M_i', \ldots, M_k\rangle]}\ \mathsf{t}_i$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N'] \quad [\mathcal{Q}, \mathcal{QV}, MN] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, MN] \to_\alpha [\mathcal{Q}', \mathcal{QV}', MN']}\ \mathsf{r.a}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M'] \quad [\mathcal{Q}, \mathcal{QV}, MN] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, MN] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M'N]}\ \mathsf{l.a}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']}{[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)] \to_\alpha [\mathcal{Q}', \mathcal{QV}', (\lambda\pi.M')]}\ \mathsf{in}.\lambda$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, U\langle r_{i_1}, ..., r_{i_n}\rangle] \in \mathcal{C} \quad \mathbf{U} : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}}{\begin{array}{c}[\mathcal{Q}, \mathcal{QV}, U\langle r_{i_1}, ..., r_{i_n}\rangle] \to_{\mathsf{Uq}} \\ [\mathbf{U}_{\langle\langle r_{i_1}, ..., r_{i_n}\rangle\rangle} Q, \mathcal{QV}, \langle r_{i_1}, ..., r_{i_n}\rangle]\end{array}}\ \mathsf{Uq}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, \mathcal{L}, \mathtt{new}(c)] \in \mathcal{C} \quad r \text{ is fresh}}{\begin{array}{c}[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \to_{\mathsf{new}} \\ [\mathcal{Q} \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, p]\end{array}}\ \mathsf{new}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \in \mathcal{C}}{\begin{array}{c}[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \to_{\mathsf{l}.\beta} \\ [\mathcal{Q}, \mathcal{QV}, M\{N/x\}]\end{array}}\ \mathsf{l}.\beta$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.M)\langle r_1, \ldots, r_n\rangle] \in \mathcal{C}}{\begin{array}{c}[\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.M)\langle r_1, \ldots, r_n\rangle] \to_{\mathsf{q}.\beta} \\ [\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}]\end{array}}\ \mathsf{q}.\beta$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, (\lambda!x.M)!N] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, (\lambda!x.M)!N] \to_{\mathsf{c}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{N/x\}]}\ \mathsf{c}.\beta$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, L((\lambda\pi.M)N)] \in \mathcal{C}}{\begin{array}{c}[\mathcal{Q}, \mathcal{QV}, L((\lambda\pi.M)N)] \to_{\mathsf{l.cm}} \\ [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.LM)N]\end{array}}\ \mathsf{l.cm}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, ((\lambda\pi.M)N)L] \in \mathcal{C}}{\begin{array}{c}[\mathcal{Q}, \mathcal{QV}, ((\lambda\pi.M)N)L] \to_{\mathsf{r.cm}} \\ [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.ML)N]\end{array}}\ \mathsf{r.cm}$$

Figure 2. Reduction rules.

$\to_{\mathscr{S}}$ by just taking the union over $\alpha \in \mathscr{S}$ of $\to_\alpha$. In particular, $\to$ will denote $\to_{\mathscr{L}}$. The usual notation for the transitive and reflexive closures will be used. In particular, $\xrightarrow{*}$ will denote the transitive and reflexive closure of $\to$.

Notice we have defined $\rightarrow$ by closing reduction rules under any context except the ones in the form $!M$. So $\rightarrow$ is not a strategy but, nevertheless, confluence holds. This is in contrast with λ_{sv}, where a strategy is indeed necessary (even if we do not take into account the nondeterministic effects of the measurement operator).

4.1 Subject Reduction

In this section we propose a Subject Reduction theorem and some related results. Notice that the calculus is type–free, so Subject Reduction is given with respect to Well Forming Rules.

Lemma 1 (Substitution Lemma (linear case))**.** *For each derivation d_1, d_2, if $d_1 \triangleright \Gamma_1, x \vdash M$ and $d_2 \triangleright \Gamma_2 \vdash N$, then $\triangleright \Gamma_1, \Gamma_2 \vdash M[N/x]$.*

Proof. The proof is by induction on the height of d_1 and by cases on the last rule. Let r be the last rule of d_1.

1. r is either *const*, or *qp–var*, or *classical–var*: trivial;
2. r is $\dfrac{\Gamma_1, x \vdash M}{\Gamma_1, !y, x \vdash M}\, weak$.
   By IH we have: $\triangleright \Gamma_1, \Gamma_2 \vdash M[N/x]$, and by means of *weak*, $\triangleright \Gamma_1, \Gamma_2, !y \vdash M[N/x]$
3. r is $\dfrac{\Gamma_1, x, y \vdash M}{\Gamma_1, x, !y \vdash M}\, der$.
   By IH we have: $\triangleright \Gamma_1, \Gamma_2, y \vdash M[N/x]$, and by means *der*: $\triangleright \Gamma_1, \Gamma_2, !y \vdash M[N/x]$
4. r is $\dfrac{\Gamma_1, x, !u, !y \vdash M}{\Gamma_1, x, !z \vdash M[z/u, z/y]}\, contr$.
   By IH we have: $\triangleright \Gamma_1, \Gamma_2, !u, !y \vdash M[N/x]$ and by means of *contr*: $\triangleright \Gamma_1, \Gamma_2, !z \vdash M[N/x][z/u, z/y]$
5. r is $\dfrac{\Gamma_1, x, y_1, \ldots, y_k \vdash M}{\Gamma_1, x, \langle y_1, \ldots, y_k \rangle \vdash M}\, Ltens$.
   By IH we have: $\triangleright \Gamma_1, \Gamma_2, y_1, \ldots, y_k \vdash M[N/x]$, and by means of *Ltens*: $\triangleright \Gamma_1, \Gamma_2, \langle y_1, \ldots, y_k \rangle \vdash M[N/x]$
6. r is $\dfrac{\Gamma_1, x \vdash M_1 \quad \Gamma_2 \vdash M_2}{\Gamma_{11}, \Gamma_{12}, x \vdash M_1 M_2}\, app$.
   By IH we have: $\triangleright \Gamma_1, \Gamma_2 \vdash M_1[N/x]$, and by means of *app*: $\triangleright \Gamma_1, \Gamma_2 \vdash M_1[N/x]$
7. r is $\dfrac{\Gamma_1 \vdash M_1 \quad \Gamma_2, x \vdash M_2}{\Gamma_{11}, \Gamma_{12}, x \vdash M_1 M_2}\, app$. As for the previous case.
8. r is $\dfrac{\Gamma_1, x, !y \vdash M}{\Gamma_1, x \vdash \lambda !y.M} \rightarrow I$.
   By IH we have: $\triangleright \Gamma_1, \Gamma_2, !y \vdash M[N/x]$, and by means of $\rightarrow I$: $\triangleright \Gamma_1, \Gamma_2 \vdash \lambda !y.M[N/x]$
9. r is $\dfrac{\Gamma_1, x, \pi \vdash M}{\Gamma_1, x \vdash \lambda \pi.M} \multimap I$. As for the previous case.
10. r is $\dfrac{\Gamma_1, x \vdash M}{\Gamma_1, x \vdash \mathtt{new}(M)}\, new$.
    By IH we have: $\triangleright \Gamma_1, \Gamma_2 \vdash M[N/x]$ and by means of *new*: $\triangleright \Gamma_1, \Gamma_2 \vdash \mathtt{new}(M[N/x])$
11. r is $\dfrac{\Gamma_{11} \vdash M_1, \ldots, \Gamma_{1i}, x \vdash M_i, \ldots, \Gamma_{1k} \vdash M_k}{\Gamma_{11}, \ldots, \Gamma_{1k}, x \vdash \langle M_1, \ldots, M_k \rangle}\, RTens$.
    By IH we have: $\triangleright \Gamma_{1i}, \Gamma_2 \vdash M_i[N/x]$, and by means of *RTens*: $\triangleright \Gamma_{11}, \ldots, \Gamma_{1k}, \Gamma_2 \vdash \langle M_1, \ldots, M_i[N/x], \ldots, M_k \rangle$
    (note that $\langle M_1, \ldots, M_i[N/x], \ldots, M_k \rangle \equiv \langle M_1, \ldots, M_k \rangle [N/x]$)

□

Lemma 2 (Substitution (non linear case))**.** *For each derivation d_1, d_2 and for every sequence (eventually empty) $x_1, \ldots, x_n$, if $d_1 \triangleright \Gamma_1, !x_1, \ldots, !x_n \vdash M$ and $d_2 \triangleright !\Gamma_2 \vdash !N$, then $\triangleright \Gamma_1, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$*

Proof. The proof is by induction on the height of d_1 and by cases on the last rule. Let r be the last rule of d_1. We use the follow notation: let $\Gamma, !\Delta \vdash !N$ be a judgment, we write $\Gamma, !\Delta^{(i)} \vdash !N^{(i)}$ to denote i–th variant, namely the judgment where we have renamed each bang variable u_j with the fresh name u_j^i. It is immediate to observe that if $d \triangleright \Gamma, !\Delta \vdash !N$, then for each i there exists a derivation d^i such that $d^i \triangleright \Gamma, !\Delta^{(i)} \vdash !N^{(i)}$, and that is, up to renaming of bang variables, identical to d.

1. r is either *const*, or *qp-var*, or *classical-var*. Then, $n = 0$, and so we obtain the result by application of weakening rule. In general we can observe that when $n = 0$, namely the sequence is empty, the results follow trivially by application of dereliction rule.
   In the following case we suppose $n \geq 0$.

2. r is $\dfrac{\Gamma_1, !x_1, \ldots, !x_n \vdash M}{\Gamma_1, !x_1, \ldots, !x_n, !x_{n+1} \vdash M}\, weak$.
   By IH, we have $\triangleright \Gamma_1, \Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$ and by means of $weak$: $\Gamma_1, \Gamma_2, !x_{n+1} \vdash M[N/x_1, \ldots, N/x_n]$

3. r is der. We must distinguish two different cases.
   In the first case r is: $\dfrac{\Gamma_1, !x_1, \ldots, !x_{n-1}, x_n}{\Gamma_1, !x_1, \ldots, !x_{n-1}, !x_n}\, der$.
   So, by IH $\triangleright \Gamma_1, !\Gamma_2, x_n \vdash M[N/x_1, \ldots, N/x_{n-1}]$ and by substitution lemma in linear case, we obtain $\triangleright \Gamma_1, !\Gamma_2, x_n \vdash M[N/x_1, \ldots, N/x_{n-1}, N/x_n]$.
   In the second case, we apply dereliction rule on a variable in Γ_1.
   Let $\Gamma_1 = \Gamma_1', y$; r is $\dfrac{\Gamma_1', y, !x_1, \ldots, !x_n \vdash M}{\Gamma_1', !y, !x_1, \ldots, !x_n \vdash M}\, der$.
   So, by IH $\triangleright \Gamma_1', y, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$. Then, by means of der $\triangleright \Gamma_1', !y, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$

4. r is $contr$; as in the previous case, we distinguish two case.
   If we contract two variables in a variable $!x_i$ not in sequence $!x_1, \ldots, !x_n$, we have $\dfrac{\Gamma_1, !x_1, \ldots, !x_n, !y, !z \vdash M}{\Gamma_1, !x_1, \ldots, !x_n, !u \vdash M}\, contr$.
   By IH we have $\triangleright \Gamma_1, !\Gamma_2, !y, !z \vdash M[N/x_1, \ldots, N/x_n]$ and applying the contraction rule on $!y$ and $!z$ we obtain $\triangleright \Gamma_1, !\Gamma_2, !u \vdash M[N/x_1, \ldots, N/x_n][u/y, u/z]$.
   Otherwise, if we contract two variables in a variable of sequence, we have $\dfrac{\Gamma_1, !x_1, \ldots, !x_{n-1}, !y, !z \vdash M}{\Gamma_1, !x_1, \ldots, !x_{n-1}, !x_n \vdash M}\, contr$.
   By IH. we have $\triangleright \Gamma_1, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_{n-1}, N/y, N/z]$ and the thesis follows observing that $M[N/x_1, \ldots, N/x_{n-1}, N/y, N/z] = M[N/x_1, \ldots, N/x_{n-1}, x_n/y, x_n/z][N/x_n]$.

5. r is $\dfrac{\Gamma_1, y_1, \ldots, y_k, !x_1, \ldots, !x_n \vdash M}{\Gamma_1, \langle y_1, \ldots, y_k \rangle, !x_1, \ldots, !x_n \vdash M}\, Ltens$.
   By IH we have $\triangleright \Gamma_1, y_1, \ldots, y_k, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$ and by means of $Ltens$ we obtain $\triangleright \Gamma_1, \langle y_1, \ldots, y_k \rangle, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$

6. r is $\dfrac{\Gamma_{11}, !x_1, \ldots, !x_k \vdash M_1 \;\; \Gamma_{12}, !x_{k+1}, \ldots, !x_n \vdash M_2}{\Gamma_{11}, \Gamma_{12}, !x_1, \ldots, !x_k, !x_{k+1}, \ldots, !x_n \vdash M_1 M_2}\, app$ and $\Gamma_1 = \Gamma_{11}, \Gamma_{12}$.
   We use IH with $!\Gamma_2^{(1)} \vdash !N^{(1)}$ and $!\Gamma_2^{(2)} \vdash !N^{(2)}$ as variants of the statement $!\Gamma_2 \vdash !N$ and we obtain $\triangleright \Gamma_{11}, !\Gamma_2^{(1)} \vdash M_1[N^{(1)}/x_1, \ldots, N^{(1)}/x_k]$ and $\triangleright \Gamma_{12}, !\Gamma_2^{(2)} \vdash M_2[N^{(2)}/x_{k+1}, \ldots, N^{(2)}/x_n]$.
   So, by means of app we have $\triangleright \Gamma_{11}, \Gamma_{12}, !\Gamma_2^{(1)}, !\Gamma_2^{(2)} \vdash M_1[N^{(1)}/x_1, \ldots, N^{(1)}/x_k] M_2[N^{(2)}/x_{k+1}, \ldots, N^{(2)}/x_n]$ and by several contractions we have thesis.

7. r is $\dfrac{\Gamma_1, !x_1, \ldots, !x_n, !y \vdash M}{\Gamma_1, !x_1, \ldots, !x_n \vdash \lambda !y.M} \to I$.
   By IH $\triangleright \Gamma_1, !\Gamma_2, !y \vdash M[N/x_1, \ldots, N/x_n]$ and by means $\to I$ we obtain $\triangleright \Gamma_1, !\Gamma_2, \vdash \lambda !y.M[N/x_1, \ldots, N/x_n]$

8. r is $\dfrac{\Gamma_1, !x_1, \ldots, !x_n, !y \vdash M}{\Gamma_1, !x_1, \ldots, !x_n \vdash \lambda !y.M} \multimap$. As for the previous case.

9. r is $\dfrac{\Gamma_1, !x_1, \ldots, !x_n \vdash M}{\Gamma_1, !x_1, \ldots, !x_n \vdash \texttt{new}(M)}\, new$.
By IH $\triangleright \Gamma_1, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$ and by means of new we obtain $\triangleright \Gamma_1, !\Gamma_2 \vdash \texttt{new}(M[N/x_1, \ldots, N/x_n])$

10. r is $\dfrac{\Gamma_{11}, !x_1, \ldots, !x_r \vdash M_1 \ldots \Gamma_{1k}, !x_s, \ldots, !x_n \vdash M_k}{\Gamma_{11}, \ldots \Gamma_{1k}, !x_1, \ldots, !x_n \vdash \langle M_1, \ldots, M_k \rangle}\, Rtens$.
We use IH with $!\Gamma_2^{(1)} \vdash !N^{(1)}, \ldots, !\Gamma_2^{(k)} \vdash !N^{(k)}$ as variants of the statement $!\Gamma_2 \vdash !N$ and we obtain
$\triangleright \Gamma_{11}, \Gamma_2^{(1)} \vdash M_1[N^{(1)}/x_1, \ldots, N^{(1)}/x_r]$
$\vdots$
$\triangleright \Gamma_{1k}, \Gamma_2^{(k)} \vdash M_1[N^{(k)}/x_s, \ldots, N^{(k)}/x_n]$
So, by means of the tensor rule we have $\triangleright \Gamma_{11}, \ldots, \Gamma_{1k}, \Gamma_2^{(1)}, \ldots, \Gamma_2^{(k)} \vdash$
$\langle M_1[N^{(1)}/x_1, \ldots, N^{(1)}/x_r], \ldots, M_k[N^{(k)}/x_s, \ldots, N^{(k)}/x_n] \rangle$, and by several application of contractions we obtain thesis.

11. r is $prom$. In order to apply the promotion rule, Γ_1 must to be $!\Delta$. Therefore r is $\dfrac{!\Delta, !x_1, \ldots, !x_n \vdash M}{!\Delta, !x_1, \ldots, !x_n \vdash !M}\, prom$.
By IH we have $\triangleright\, !\Delta, !\Gamma_2 \vdash M[N/x_1, \ldots, N/x_n]$ and by means of $prom$ $\triangleright\, !\Delta, !\Gamma_2 \vdash !M[N/x_1, \ldots, N/x_n]$

□

Lemma 3 (Substitution (quantum case))**.** *For each derivation d_1, d_2, for every non empty sequence $x_1, \ldots, x_n$, and for every non empty sequence $r_1, \ldots, r_n$,*
if $d_1 \triangleright \Gamma_1, \langle x_1, \ldots, x_n \rangle \vdash M$ and $d_2 \triangleright\, !\Gamma_2, r_1, \ldots, r_n \vdash \langle r_1, \ldots, r_n \rangle$,
with $r_1, \ldots, r_n \notin M$, then $\triangleright \Gamma_1, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$

Proof. The proof is by induction on the height of d_1, and by cases on the last rule. Let r be the last rule of d_1.

1. r is either *const*, or *qp-var*, or *classical-var*: trivial;

2. r is $\dfrac{\Gamma_1, \langle x_1, \ldots, x_n \rangle \vdash M}{\Gamma_1, !y, \langle x_1, \ldots, x_n \rangle \vdash M}\, weak$.
By IH we have: $\triangleright \Gamma_1, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$ and by means of *weak*,
$\triangleright \Gamma_1, !y, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$

3. r is $\dfrac{\Gamma_1, !u, !y, \langle x_1, \ldots, x_n \rangle \vdash M}{\Gamma_1, !z, \langle x_1, \ldots, x_n \rangle \vdash M}\, contr$.
By IH we have: $\triangleright \Gamma_1, !u, !y, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$, and by means of *contr*:
$\triangleright \Gamma_1, !z, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$

4. r is $\dfrac{\Gamma_1, x_1, \ldots, x_n \vdash M}{\Gamma_1, \langle x_1, \ldots, x_n \rangle \vdash M}\, Ltens$.
By means of lemma1, applied to $\triangleright \Gamma_1, x_1, \ldots, x_n \vdash M$ and $r_1 \vdash r_1$ we obtain $\Gamma_1, r_1, x_2, \ldots, x_n \vdash M[r_1/x_1]$, and by successive applications of the lemma 1 with respect axioms $r_2 \vdash r_2, \ldots, r_n \vdash r_n$ we obtain
$\triangleright \Gamma_1, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$.
Then, by several application of weakening, we obtain $\triangleright \Gamma_1, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$

5. Let us suppose that the pattern $\langle x_1, \ldots, x_n \rangle$ belong to the left judgment of the rule r (the symmetric case is handled in a similar way).
r is $\dfrac{\Gamma_{11}, \langle x_1, \ldots, x_n \rangle \vdash M_1 \quad \Gamma_{12} \vdash M_2}{\Gamma_{11}, \Gamma_{12}, \langle x_1, \ldots, x_n \rangle \vdash M_1 M_2}\, app$.
By IH we have $\triangleright \Gamma_{11}, !\Gamma_2, r_1, \ldots, r_n \vdash M_1[r_1/x_1, \ldots, r_n/x_n]$ and by means of *app*:

$\Gamma_{11}, \Gamma_{12}, !\Gamma_2, r_1, \ldots, r_n \vdash M_1[r_1/x_1, \ldots, r_n/x_n]M_2$
Observe that $M_1[r_1/x_1, \ldots, r_n/x_n]M_2 \equiv (M_1M_2)[r_1/x_1, \ldots, r_n/x_n]$ and conclude.

6. r is $\dfrac{\Gamma_1, \langle x_1, \ldots, x_n\rangle, !y \vdash M}{\Gamma_1, \langle x_1, \ldots, x_n\rangle \vdash \lambda !y.M} \to I$.
By IH we have
$\triangleright\, \Gamma_1, !y, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$ and by means of $\to I$:
$\Gamma_1, !\Gamma_2, r_1, \ldots, r_n \vdash \lambda !y.M[r_1/x_1, \ldots, r_n/x_n]$

7. r is $\dfrac{\Gamma_1, \langle x_1, \ldots, x_n\rangle, \pi \vdash M}{\Gamma_1, \langle x_1, \ldots, x_n\rangle \vdash \lambda \pi.M} \multimap I$. As for the previous case.

8. r is $\dfrac{\Gamma_1, \langle x_1, \ldots, x_n\rangle \vdash M}{\Gamma_1, \langle x_1, \ldots, x_n\rangle \vdash \mathtt{new}(M)}\, new$.
By IH we have $\triangleright\, \Gamma_1, !\Gamma_2, r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$ and by means of new rule we obtain
$\triangleright\, \Gamma_1, !\Gamma_2, r_1, \ldots, r_n \vdash \mathtt{new}(M[r_1/x_1, \ldots, r_n/x_n])$

9. r is $\dfrac{\Gamma_{11} \vdash M_1 \ \ldots\ \Gamma_{1i}, \langle x_1, \ldots, x_n\rangle \vdash M_i \ \ldots\ \Gamma_{1k} \vdash M_k}{\Gamma_{11}, \ldots, \Gamma_{1k}, \langle x_1, \ldots, x_n\rangle \vdash \langle M_1, \ldots, M_k\rangle}\, Rtens$.
By IH (w.r.t. $\triangleright\, \Gamma_{1i}, \langle x_1, \ldots, x_n\rangle \vdash M_i$) we have
$\triangleright\, \Gamma_{1i}, !\Gamma_2, r_1, \ldots, r_n \vdash M_1[r_1/x_1, \ldots, r_n/x_n]$ and by means of $Rtens$ we conclude
$\triangleright\, \Gamma_{11}, \ldots, \Gamma_{1k}, !\Gamma_2, r_1, \ldots r_n \vdash \langle M_1, \ldots, M_i[r_1/x_1, \ldots, r_n/x_n], \ldots, M_k\rangle$,
(observe that $\langle M_1, \ldots, M_i[r_1/x_1, \ldots, r_n/x_n], \ldots, M_k\rangle \equiv \langle M_1, \ldots, M_k\rangle[r_1/x_1, \ldots, r_n/x_n]$).

Note that last rule can't be a promotion rule.

□

Theorem 1 (Subject Reduction)**.** *If* $\triangleright\, \Gamma \vdash M$ *and* $[\mathcal{Q}, \mathcal{QV}, M] \to [\mathcal{Q}', \mathcal{QV}', M']$ *then* $\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M'$.

Proof. The proof is by induction on the height of d and by cases on the last rule of d. Let r be the last rule of d.

1. r is either *const*, or *qp-var*, or *classical-var*: the proof is trivial.
2. r is app and the transition rule is $\dfrac{[\mathcal{Q}, \mathcal{QV}, M_1] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M_1'] \quad [\mathcal{Q}, \mathcal{QV}, M_1M_2] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, M_1M_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M_1'M_2]}$ l.a
We have $\dfrac{\Gamma_1 \vdash M_1' \quad \Gamma_2 \vdash M_2}{\Gamma_1, \Gamma_2 \vdash M_1'M_2}\, app$, so by IH we have $\triangleright\ \Gamma_1, \mathcal{QV}' - \mathcal{QV} \vdash M_1'$, and by means of app we obtain
$\triangleright\ \Gamma_1, \Gamma_2, \mathcal{QV}' - \mathcal{QV} \vdash M_1'M_2$.
3. r is app and the transition rule $\dfrac{[\mathcal{Q}, \mathcal{QV}, M_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M_2'] \quad [\mathcal{Q}, \mathcal{QV}, M_1M_2] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, M_1M_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M_1M_2']}$ r.a : simmetric to previous case.
4. r is app and the transition rule is $\dfrac{[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \to_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{N/x\}]}$ l.β (application generates a redex).
Suppose we have the follow derivation d:

$$\dfrac{\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Gamma_1, x \vdash M\end{array}}{\Gamma_1 \vdash \lambda x.M} \qquad \begin{array}{c}d_2\\ \vdots\\ \Gamma_2 \vdash N\end{array}}{\Gamma_1, \Gamma_2 \vdash (\lambda x.M)N}$$

Let $d_1 \triangleright\ \Gamma_1, x \vdash M$ and $d_2 \triangleright\ \Gamma_2 \vdash N$.
Considering the transition $[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \to_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, M[N/x]]$.
We note that the transition doesn't modify $\mathcal{QV}$ set, so we've just to apply substitution lemma on d_1 and d_2: $\triangleright\, \Gamma_1, \Gamma_2 \vdash M[N/x]$.

5. r is app and the transition rule is $\mathsf{q}.\beta$ or $\mathsf{c}.\beta$. Similar to previous case.

6. r is app and the transition rule is $\dfrac{[\mathcal{Q}, \mathcal{QV}, L((\lambda p.M)N)] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, L((\lambda\pi.M)N)] \rightarrow_{\mathsf{l.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.LM)N]}\ \mathsf{l.cm}$.

   Note that the transition rule doesn't modify $\mathcal{Q}$ and $\mathcal{QV}$. So, from derivation:

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma_1 \vdash L\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma_2', \pi \vdash M\end{array}}{\Gamma_2' \vdash \lambda\pi.M} \multimap I \qquad \begin{array}{c}\vdots\\ \Gamma_2'' \vdash N\end{array}}{\Gamma_2 \vdash (\lambda\pi.M)N}\, app}{\Gamma_1, \Gamma_2 \vdash L((\lambda\pi.M)N)}\, app$$

   we exhibit a derivation of $\Gamma_1, \Gamma_2 \vdash (\lambda\pi.LM)N$:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma_1 \vdash L\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma_2', \pi \vdash M\end{array}}{\Gamma_1, \Gamma_2', \pi \vdash LM}\, app}{\Gamma_1, \Gamma_2' \vdash \lambda\pi.LM} \multimap I \qquad \begin{array}{c}\vdots\\ \Gamma_2'' \vdash N\end{array}}{\Gamma_1, \Gamma_2 \vdash (\lambda\pi.LM)N}\, app$$

7. r is app and the transition rule is $\dfrac{[\mathcal{Q}, \mathcal{QV}, ((\lambda p.M)N)L] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, ((\lambda\pi.M)N)L] \rightarrow_{\mathsf{r.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.ML)N]}\ \mathsf{r.cm}$.

   As in previous case,

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma_1', \pi \vdash M\end{array}}{\Gamma_1' \vdash \lambda\pi.M} \multimap I \qquad \begin{array}{c}\vdots\\ \Gamma_1'' \vdash N\end{array}}{\Gamma_1 \vdash (\lambda\pi.M)N}\, app \qquad \begin{array}{c}\vdots\\ \Gamma_2 \vdash L\end{array}}{\Gamma_1, \Gamma_2 \vdash ((\lambda\pi.M)N)L}\, app$$

   then

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma_1', \pi \vdash M\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma_2 \vdash L\end{array}}{\Gamma_1', \Gamma_2, \pi \vdash ML}\, app}{\Gamma_1', \Gamma_2 \vdash \lambda\pi.ML} \multimap I \qquad \begin{array}{c}\vdots\\ \Gamma_1'' \vdash N\end{array}}{\Gamma_1, \Gamma_2 \vdash ((\lambda\pi.ML)N)}\, app$$

8. r is $\multimap I$:

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, \pi \vdash M\end{array}}{\Gamma \vdash \lambda\pi.M}$$

   If we have

$$\dfrac{[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']}{[\mathcal{Q}, \mathcal{QV}, \lambda\pi.M] \rightarrow [\mathcal{Q}', \mathcal{QV}', \lambda\pi.M']}$$

by IH on d_1

$$\triangleright\ \Gamma, \pi, \mathcal{QV}' - \mathcal{QV} \vdash M'$$

and we conclude

$$\triangleright\ \Gamma, \pi, \mathcal{QV}' - \mathcal{QV} \vdash \lambda.\pi M'$$

9. r is

$$\frac{!\Gamma \vdash c}{!\Gamma \vdash \mathtt{new}(c)}$$

We have the following transition rule:

$$[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \rightarrow [\mathcal{Q} \otimes |p \leftarrow c\rangle, \mathcal{QV} \cup \{p\}, p]$$

Beginning from axiom

$$\frac{}{p \vdash p}$$

we obtain the result by several application of weakening rule

$$\begin{array}{c} p \vdash p \\ \vdots \\ !\Gamma, p \vdash p \end{array}$$

10. r is

$$\frac{!\Gamma \vdash M}{!\Gamma \vdash \mathtt{new}(M)}$$

in which the argument is a term M. In this case the proof is in practice identical to the case of application.

11. r is $\dfrac{\Gamma_1 \vdash M_1 \cdots \Gamma_k \vdash M_k}{\Gamma_1, \ldots, \Gamma_k \vdash \langle M_1, \ldots, M_k \rangle}\, Rtens$ and the transition rule is :

$$\frac{[\mathcal{Q}, \mathcal{QV}, M_i] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', M_i']\ [\mathcal{Q}, \mathcal{QV}, \langle M_1, \ldots, M_i, \ldots, M_k \rangle] \in \mathcal{C}}{[\mathcal{Q}, \mathcal{QV}, , \langle M_1, \ldots, M_i, \ldots, M_k \rangle] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', , \langle M_1, \ldots, M_i', \ldots, M_k \rangle]}$$

Then, if we have $d_1 \triangleright\ \Gamma_1, \ldots, d_k \triangleright\ \Gamma_k$

by HI, $\triangleright\ \Gamma_i, \mathcal{QV}' - QV \vdash M_i$ and by means of $Rtens \triangleright\ \Gamma_1, \ldots, \mathcal{QV}' - \mathcal{QV}, \ldots, \Gamma_k \vdash M_1, \ldots, M_k$

□

Corollary 1. *If* $\triangleright \Gamma \vdash M$ *and* $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ *then* $\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M$.

Notice that $\mathcal{QV}' - \mathcal{QV}$ is the (possibly empty) set of new quantum variables added to $\mathcal{QV}$ by the reduction.
In the following, we will work with *well–formed* configuration:

Definition 6. A configuration $[\mathcal{Q}, \mathcal{QV}, M]$ is said to be *well–formed* if there is a context Γ such that $\Gamma \vdash M$ is well-formed.

As a consequence of Subject Reduction, the set of well–formed configurations is closed under reduction:

Corollary 2. *If* M *is well formed and* $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ *then* M' *is well formed.*

In the following, with *configuration* we will mean *well–formed configuration*. Now, let us give the definitions of normal form, configuration and computation.

Definition 7. A configuration $C = [\mathcal{Q}, \mathcal{QV}, M]$ is said to be in *normal form* iff there is no C' such that $C \rightarrow C'$. Let us denote with NF the set of configurations in normal form.

We define a computation as a suitable sequence of configurations:

Definition 8. Let $C = [\mathcal{Q}, \mathcal{QV}, M]$ be a configuration. A *computation* of length $\varphi \leq \omega$ starting with C_0 is a sequence of configurations $\{C_i\}_{i<\varphi}$ such that for all $0 < i < \varphi$, $C_{i-1} \rightarrow C_i$ and either $\varphi = \omega$ or $C_{l-1} \in \mathsf{NF}$.

If a computation starts with a configuration $[\mathcal{Q}_0, \mathcal{QV}_0, M_0]$ in which M_0 does not contain quantum variables and the set $\mathcal{QV}_0$ is empty, then at each step i the set $\mathcal{QV}_i$ coincides with the set $\mathbf{Q}(M_i)$:

Proposition 2. *Let $\{[\mathcal{Q}_i, \mathcal{QV}_i, M_i]\}_{i<l}$ be a computation, such that $\mathbf{Q}(M_0) = \emptyset$. Then for every $i < \varphi$ we have $\mathcal{QV}_i = \mathbf{Q}(M_i)$.*

Proof. Observe that if $[\mathcal{Q}, \mathbf{Q}(M), M] \to [\mathcal{Q}', \mathcal{QV}', M']$ then by inspection of reduction rules we immediately have that $\mathcal{QV}' = \mathbf{Q}(M')$, and conclude. □

In the rest of the paper, $[\mathcal{Q}, M]$ denotes the configuration $[\mathcal{Q}, \mathbf{Q}(M), M]$.

4.2 Confluence

In this section we will work with both preconfigurations and configurations (in particular with preconfiguration we mean well–formed preconfiguration, where the notion of well–formed preconfiguration is the same of well–formed configuration). With C we denote the set of well–formed preconfigurations.

The reduction relation $C \to_\alpha C'$ between preconfigurations is defined as for configurations.

If C, C' are configurations (remember that they are equivalence classes) such that $C \to_\alpha C'$ and $C^* \in C$ is a preconfiguration, then there is a preconfiguration $C^o \in C'$ s.t. $C^* \to_\alpha C^o$. On the other side if C^*, C^o are preconfigurations such that $C^* \to_\alpha C^o$, then $C \to_\alpha C'$ where C, C' are the equivalence classes respectively of C^*, C^o.

Commutative reduction steps behave very differently to other reduction steps when considering confluence. As a consequence, it is useful to define two subsets of $\mathscr{L}$ as follows:

Definition 9. We distinguish two particular subsets of $\mathscr{L}$, namely $\mathscr{O} = \{\mathsf{r.cm}, \mathsf{l.cm}\}$ and $\mathscr{N} = \mathscr{L} - \mathscr{O}$.

The following two lemmas refer to preconfigurations.

Lemma 4 (Uniformity)**.** *For every M, M' such that $M \to_\alpha M'$, exactly one of the following conditions holds:*

1. *$\alpha \neq \mathsf{new}$ and there is a unitary transformation $G_{M,M'} : \mathcal{H}(\mathbf{Q}(M)) \to \mathcal{H}(\mathbf{Q}(M))$ such that $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']$ iff $[\mathcal{Q}, \mathcal{QV}, M] \in \mathsf{C}$, $\mathcal{QV}' = \mathcal{QV}$ and $\mathcal{Q}' = (G_{M,M'} \otimes I_{\mathcal{QV}-\mathbf{Q}(M)})\mathcal{Q}$.*
2. *$\alpha = \mathsf{new}$ and there are a constant c and a quantum variable r such that $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathsf{new}} [\mathcal{Q}', \mathcal{QV}', M']$ iff $[\mathcal{Q}, \mathcal{QV}, M] \in \mathsf{C}$, $\mathcal{QV}' = QV \cup \{r\}$ and $\mathcal{Q}' = Qr \otimes |r \leftarrow c\rangle$. Moreover, $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathsf{new}} [\mathcal{Q} \otimes |r' \leftarrow c\rangle, \mathcal{QV} \cup \{r'\}, M'\{r'/r\}]$ whenever $[\mathcal{Q}, \mathcal{QV}, M] \in \mathsf{C}$ and $r' \notin \mathcal{QV}$.*

Proof. We go by induction on M. M cannot be a variable nor a constant nor a unitary operator. If M is an abstraction $\lambda\pi.N$, then $M' = \lambda\pi.N'$, $N \to_\alpha N'$ and the thesis follows from the inductive hypothesis. Similarly when $M = \lambda!x.N$. If $M = NL$, then we distinguish a number of cases:

- $M' = N'L$ and $N \to_\alpha N'$. The thesis follows from the inductive hypothesis.
- $M' = NL'$ and $L \to_\alpha L'$. The thesis follows from the inductive hypothesis.
- $N = U^n$, $L = \langle r_{i_1}, ..., r_{i_n}\rangle$ and $M' = \langle r_{i_1}, ..., r_{i_n}\rangle$. Then case 1 holds. In particular, $\mathbf{Q}(M) = \{r_{i_1}, ..., r_{i_n}\}$ and $G_{M,M'} = U_{r_{i_1},...,r_{i_n}}$.
- $N = \lambda x.P$ and $M' = P\{L/x\}$. Then case 1 holds. In particular $G_{M,M'} = I_{\mathbf{Q}(M)}$.
- $N = \lambda\langle x_1, \ldots, x_n\rangle.P$, $L = \langle r_1, \ldots, r_n\rangle$ and $M' = P\{r_1/x_1, \ldots, r_n/x_n\}$. Then case 1 holds and $G_{M,M'} = I_{\mathbf{Q}(M)}$.
- $N = \lambda!x.P$, $L =!Q$ and $M' = P\{Q/x\}$. Then case 1 holds and $G_{M,M'} = I_{\mathbf{Q}(M)}$.
- $L = (\lambda\pi.P)Q$ and $M' = (\lambda\pi.NP)Q$. Then case 1 holds and $G_{M,M'} = I_{\mathbf{Q}(M)}$.
- $N = (\lambda\pi.P)Q$ and $M' = (\lambda\pi.PL)Q$. Then case 1 holds and $G_{M,M'} = I_{\mathbf{Q}(M)}$.

If $M = \mathtt{new}(c)$ then M' is a quantum variable r and case 2 holds. This concludes the proof. □

Lemma 5. *Suppose $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']$.*

1. *If $[[\mathcal{Q}, \mathcal{QV}, M\{N/x\}]] \in \mathsf{C}$, then $[\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M'\{N/x\}]$.*
2. *If $[[\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}]] \in \mathsf{C}$, then $[\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M\{r_1/x_1, \ldots, r_n/x_n\}]$*
3. *If $x, \Gamma \vdash N$ and $[[\mathcal{Q}, \mathcal{QV}, N\{M/x\}]] \in \mathsf{C}$, then $[\mathcal{Q}, \mathcal{QV}, N\{M/x\}] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N\{M'/x\}]$*

Proof. Claims 1 and 2 can be proved by induction on the proof of $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']$. Claim 3 can be proved by induction on N. □

Strictly speaking, one-step confluence does not hold in the Q-calculus. For example, if $[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \in \mathcal{C}$, then both

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \to_{\mathscr{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{x/L\})]$$

and

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \to_{\mathscr{O}} [\mathcal{Q}, \mathcal{QV}, (\lambda x.(\lambda\pi.M)N)L] \to_{\mathscr{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{x/L\})]$$

However, this phenomenon is only due to the presence of commutative rules:

Lemma 6 (One-step Confluence for preconfigurations)**.** *Let C, D, E be preconfigurations with $C \to_\alpha D$, $C \to_\beta E$ and $D \neq E$.*

1. *If $\alpha \in \mathscr{O}$ and $\beta \in \mathscr{O}$, then there is F with $D \to_{\mathscr{O}} F$ and $E \to_{\mathscr{O}} F$.*
2. *If $\alpha \in \mathscr{N}$ and $\beta \in \mathscr{N}$, then there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{N}} F$.*
3. *If $\alpha \in \mathscr{O}$ and $\beta \in \mathscr{N}$, then either $D \to_{\mathscr{N}} E$ or there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{O}} F$.*

Proof. Let $C = [\mathcal{Q}, QV, M]$. We go by induction on M. M cannot be a variable nor a constant nor a unitary operator. If M is an abstraction $\lambda\pi.N$, then $D = [\mathcal{Q}', \mathcal{QV}', \lambda\pi.N']$, $D' = [\mathcal{Q}'', \mathcal{QV}'', \lambda\pi.N'']$ and

$$\begin{aligned} [\mathcal{Q}, \mathcal{QV}, N] &\to_\alpha [\mathcal{Q}', \mathcal{QV}', N'] \\ [\mathcal{Q}, \mathcal{QV}, N] &\to_\beta [\mathcal{Q}'', \mathcal{QV}'', N''] \end{aligned}$$

The induction hypothesis easily leads to the thesis. If $M = NL$, we can distinguish a number of distinct cases depending on the last rule used to prove $C \to_\alpha D$, $C \to_\beta E$:

- $D = [\mathcal{Q}', \mathcal{QV}', N'L]$ and $E = [\mathcal{Q}'', \mathcal{QV}'', NL']$ where $[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N']$ and $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{Q}'', \mathcal{QV}'', L']$. We need to distinguish four sub-cases:
  - If $\alpha, \beta = \mathsf{new}$, then, by Lemma 4, there exist two quantum variables $r', r'' \notin \mathcal{QV}$ and two constants c', c'' such that $\mathcal{QV}' = \mathcal{QV} \cup \{r'\}$, $\mathcal{QV}'' = \mathcal{QV} \cup \{r''\}$, $\mathcal{Q}' = \mathcal{Q} \otimes |r' \leftarrow c'\rangle$ and $\mathcal{Q}'' = \mathcal{Q} \otimes |r'' \leftarrow c''\rangle$. Applying 4 again, we obtain

$$\begin{aligned} D &\to_{\mathsf{new}} [\mathcal{Q} \otimes |r' \leftarrow c'\rangle \otimes |r''' \leftarrow c''\rangle, \mathcal{QV} \cup \{r', r'''\}, N'L'\{r'''/r''\}] = F \\ E &\to_{\mathsf{new}} [\mathcal{Q} \otimes |r'' \leftarrow c''\rangle \otimes |r'''' \leftarrow c'\rangle, \mathcal{QV} \cup \{r'', r''''\}, N'\{r''''/r'\}L'] = G \end{aligned}$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha = \mathsf{new}$ and $\beta \neq \mathsf{new}$, then, by Lemma 4 there exists a quantum variable r and a constant c such that $\mathcal{QV}' = \mathcal{QV} \cup \{r\}$, $\mathcal{Q}' = \mathcal{Q} \otimes |r \leftarrow c\rangle$, $\mathcal{QV}'' = \mathcal{QV}$ and $\mathcal{Q}'' = (G_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}$. As a consequence, applying Lemma 4 again, we obtain

$$\begin{aligned} D &\to_\beta [(G_{L,L'} \otimes I_{\mathcal{QV} \cup \{r\} - \mathbf{Q}(L)})(\mathcal{Q} \otimes |r \leftarrow c\rangle), \mathcal{QV} \cup \{r\}, N'L'] = F \\ E &\to_{\mathsf{new}} [((G_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}) \otimes |r \leftarrow c\rangle, \mathcal{QV} \cup \{r\}, N'L'] = G \end{aligned}$$

    As can be easily checked, $F = G$.
  - If $\alpha \neq \mathsf{new}$ and $\beta = \mathsf{new}$, then we can proceed as in the previous case.
  - If $\alpha, \beta \neq \mathsf{new}$, then by Lemma 4, there exist $\mathcal{QV}'' = \mathcal{QV}' = \mathcal{QV}$, $\mathcal{Q}' = (G_{N,N'} \otimes I_{\mathcal{QV} - \mathbf{Q}(N)})\mathcal{Q}$ and $\mathcal{Q}'' = (G_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}$. Applying 4 again, we obtain

$$\begin{aligned} D &\to_\beta [(G_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})((G_{N,N'} \otimes I_{\mathcal{QV} - \mathbf{Q}(N)})\mathcal{Q}), \mathcal{QV}, N'L'] = F \\ E &\to_\alpha [(G_{N,N'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})((G_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}), \mathcal{QV}, N'L'] = G \end{aligned}$$

    As can be easily checked, $F = G$.
- $D = [\mathcal{Q}', \mathcal{QV}', N'L]$ and $E = [\mathcal{Q}'', \mathcal{QV}'', N''L]$ where $[\mathcal{Q}, QV, N] \to [\mathcal{Q}', QV', N']$ and $[\mathcal{Q}, \mathcal{QV}, N] \to [\mathcal{Q}'', \mathcal{QV}'', N'']$. Here we can apply the inductive hypothesis.
- $D = [\mathcal{Q}', \mathcal{QV}', NL']$ and $E = [\mathcal{Q}'', \mathcal{QV}'', NL'']$ where $[\mathcal{Q}, QV, L] \to [\mathcal{Q}', QV', L']$ and $[\mathcal{Q}, \mathcal{QV}, L] \to [\mathcal{Q}'', \mathcal{QV}'', L'']$. Here we can apply the inductive hypothesis as well.

- $N = (\lambda x.P)$, $D = [\mathcal{Q}, \mathcal{QV}, P\{L/x\}]$, $E = [\mathcal{Q}', \mathcal{QV}', NL']$, where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{Q}', \mathcal{QV}', L']$.
  Clearly $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \in \mathcal{C}$ and, by Lemma 5, $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \to [\mathcal{Q}', \mathcal{QV}', P\{L'/x\}]$.
  Moreover, $[\mathcal{Q}', \mathcal{QV}', NL'] = [\mathcal{Q}', \mathcal{QV}', (\lambda x.P)L'] \to [\mathcal{Q}', \mathcal{QV}', P\{L'/x\}]$
- $N = (\lambda x.P)$, $D = [\mathcal{Q}, \mathcal{QV}, P\{L/x\}]$, $E = [\mathcal{Q}', \mathcal{QV}', (\lambda x.P')L]$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \in \mathcal{C}$ and, by Lemma 5, $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{L/x\}]$.
  Moreover, $[\mathcal{Q}', \mathcal{QV}', (\lambda x.P')L] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{L/x\}]$
- $N = (\lambda !x.P)$, $L = !Q$, $D = [\mathcal{Q}, \mathcal{QV}, P\{Q/x\}]$, $E = [\mathcal{Q}', \mathcal{QV}', (\lambda !x.P')L]$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{Q/x\}] \in \mathcal{C}$ and, by Lemma 5, $[\mathcal{Q}, \mathcal{QV}, P\{Q/x\}] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{Q/x\}]$.
  Moreover, $[\mathcal{Q}', \mathcal{QV}', (\lambda x.P')!Q] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{Q/x\}]$
- $N = (\lambda \langle x_1, \ldots, x_n \rangle .P)$, $L = \langle r_1, \ldots, r_n \rangle$, $D = [\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}]$, $E = [\mathcal{Q}', \mathcal{QV}', (\lambda \langle x_1, \ldots, x_n \rangle .P')L]$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathcal{C}$ and, by Lemma 5, $[\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{r_1/x_1, \ldots, r_n/x_n\}]$.
  Moreover, $[\mathcal{Q}', \mathcal{QV}', (\lambda \langle x_1, \ldots, x_n \rangle .P')L] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{r_1/x_1, \ldots, r_n/x_n\}]$.
- $N = (\lambda x.P)Q$, $D = [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E = [\mathcal{Q}, \mathcal{QV}, (P\{Q/x\})L]$, $\alpha = \mathsf{r.cm}$, $\beta = \mathsf{l}.\beta$.
  Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, (P\{Q/x\})L]$.
- $N = (\lambda \pi.P)Q$, $D = [\mathcal{Q}, \mathcal{QV}, (\lambda \pi.PL)Q]$, $E = [\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P')Q)L]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$.
  Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.P'L)Q]$ and $[\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P')Q)L] \to_\beta [\mathcal{Q}', \mathcal{QV}', (\lambda \pi.P'L)Q]$.
- $N = (\lambda \pi.P)Q$, $D = [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E = [\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q')L]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, Q] \to_\beta [\mathcal{Q}', \mathcal{QV}', Q']$.
  Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.PL)Q']$ and $[\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q')L] \to_\beta [\mathcal{Q}', \mathcal{QV}', (\lambda \pi.PL)Q']$.
- $N = (\lambda \pi.P)Q$, $D = [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E = [\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q)L']$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{Q}', \mathcal{QV}', L']$.
  Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.PL')Q]$ and $[\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q)L'] \to_\beta [\mathcal{Q}', \mathcal{QV}', (\lambda \pi.PL')Q]$.
- $N = (\lambda \pi.P)$, $L = (\lambda x.Q)R$, $D = [\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R]$, $E = [\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$, $\alpha = \mathsf{l.cm}$, $\beta = \mathsf{l}.\beta$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R] \to \mathsf{l}.\beta[\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$.

M cannot be in the form $\mathtt{new}(c)$, because in that case $D \equiv E$. This concludes the proof. □

As a simple corollary of the previous lemma we have the following one–step confluence property for configurations:

Proposition 3 (One-step Confluence)**.** *Let C, D, E be configurations with $C \to_\alpha D$, $C \to_\beta E$ and $D \neq E$.*
1. *If $\alpha \in \mathscr{O}$ and $\beta \in \mathscr{O}$, then there is F with $D \to_{\mathscr{O}} F$ and $E \to_{\mathscr{O}} F$.*
2. *If $\alpha \in \mathscr{N}$ and $\beta \in \mathscr{N}$, then there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{N}} F$.*
3. *If $\alpha \in \mathscr{O}$ and $\beta \in \mathscr{N}$, then either $D \to_{\mathscr{N}} E$ or there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{O}} F$.*

The fact a strong confluence result like Proposition 3 holds here is a consequence of having adopted the so-called *surface reduction*: it is not possible to reduce inside a subterm in the form $!M$ and, as a consequence, it is not possible to erase a diverging term. This has been already pointed out by Simpson [16].

Even in absence of types, we cannot build an infinite sequence of commuting reductions:

Lemma 7. *The relation $\to_{\mathscr{O}}$ is strongly normalizing. In other words, there cannot be any infinite sequence $C_1 \to_{\mathscr{O}} C_2 \to_{\mathscr{O}} C_3 \to_{\mathscr{O}} \ldots$.*

Proof. Define the size $|M|$ of a term M as the number of symbols in it. Moreover, define the abstraction size $|M|_\lambda$ of M as the sum over all subterms of M in the form $\lambda \pi.N$, of $|N|$. Clearly $|M|_\lambda \leq |M|^2$. Moreover, if $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathscr{O}} [\mathcal{Q}, \mathcal{QV}, M']$, then $|M'| = |M|$ but $|M'|_\lambda > |M|_\lambda$. This concludes the proof. □

The following definition is useful when talking about reduction lengths, and takes into account both commuting and non-commuting reductions:

Definition 10. Let $C_1, \ldots, C_n$ be a sequence of (pre)configurations such that $C_1 \to \ldots \to C_n$. The sequence is called an *m-sequence of length n from C_1 to C_n* iff m is a natural number and there is $A \subseteq \{1, \ldots, n-1\}$ with $|A| = m$ and $C_i \to_{\mathscr{N}} C_{i+1}$ iff $i \in A$. If there is a m-sequence of length n from C to C', we will write $C \xrightarrow{m,n} C'$ or simply $C \xrightarrow{m} C'$.

Lemma 8. *Let C, C', D be preconfigurations with $C \equiv C'$ and $C \to_\alpha D$. Then there is $D' \equiv D$ with $C' \to_\alpha D'$.*

Proof. Let $C = [\mathcal{Q}, \mathcal{QV}, M]$. We go by induction on M. □

Proposition 4. *Let C, C', D, D' be preconfigurations with $C \to_\alpha D$, $C' \to_\beta D'$. Then exactly one of the following conditions hold:*

1. *There are E, E' with $E \equiv E'$ such that $D \to_\beta E$, $D' \to_\alpha E'$.*
2. *$\alpha \in \mathscr{O}$, $\beta \in \mathscr{N}$ and there is E with $E \equiv D'$ such that $D \to_\beta E$*
3. *$\alpha \in \mathscr{N}$, $\beta \in \mathscr{O}$ and there is E with $E \equiv D$ such that $D' \to_\alpha E$.*

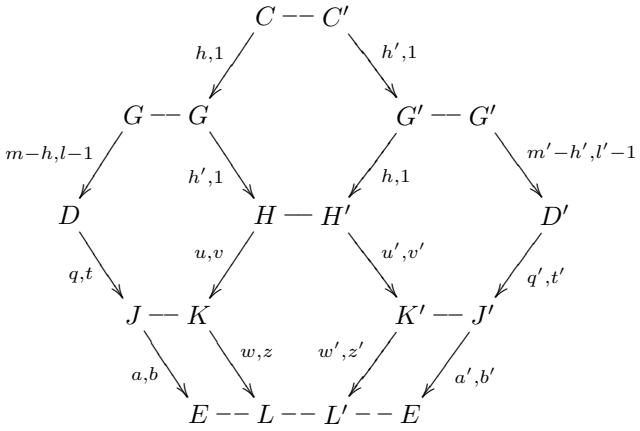
Proof. An easy corollary of Lemma 6 and Lemma 8. □

This way we can generalize Lemma 6 to another one talking about reduction sequences of arbitrary length:

Proposition 5. *Let C, D, D' be preconfigurations with $C \xrightarrow{m} D$ and $C \xrightarrow{m'} D'$. Then, there are preconfigurations E, E' with $E \equiv E'$, $D \xrightarrow{n} E$ and $D' \xrightarrow{n'} E'$ with $n \leq m'$, $n' \leq m$ and $n + m = n' + m'$.*

Proof. We prove the following, stronger statement: suppose there are C, C', D, D' with $C \equiv C'$, a m-sequence of length l from C to D and an m'-sequence of length l' from C' to D'. Then, there are a preconfiguration E, E' with $E \equiv E'$, a n-sequence of length k from D to E and n'-sequence of length k' from D' to E' with $n \leq m'$, $n' \leq m$, $k \leq l'$, $k' \leq l$ and $n + m = n' + m'$. We go by induction on $l + l'$. If $l + l' = 0$, then $C = D$, $C' = D'$, $E = D$, $E' = D'$ and all the involved natural numbers are 0. If $l = 0$, then $D = C$, $E' = D'$ and E can be obtained applying l' times Lemma 8. Similarly when $l' = 0$. So, we can assume $l, l' > 0$. There are G, G', two integers $h, h' \leq 1$ with $C \to_\alpha G'$ and $C' \to_\beta G'$, an $(m - h)$-sequence of length $l - 1$ from G to D and an $(m' - h')$-sequence of length $l' - 1$ from G' to D' We can distinguish three cases, depending on the outcome of Proposition 4:

- There are H, H' with $G \to_\beta H$ and $G' \to_\alpha H'$. By applying several times the induction hypothesis, we end up with the following diagram



together with the equations:

$$\begin{array}{llll} q \leq h' & q' \leq h & w \leq u' & a = w \\ t \leq 1 & t' \leq 1 & z \leq v' & b \leq z \\ u \leq m - h & u' \leq m' & w' \leq u & a' = w' \\ v \leq l - 1 & v' \leq l' - 1 & z' \leq v & b' \leq z' \end{array}$$

and

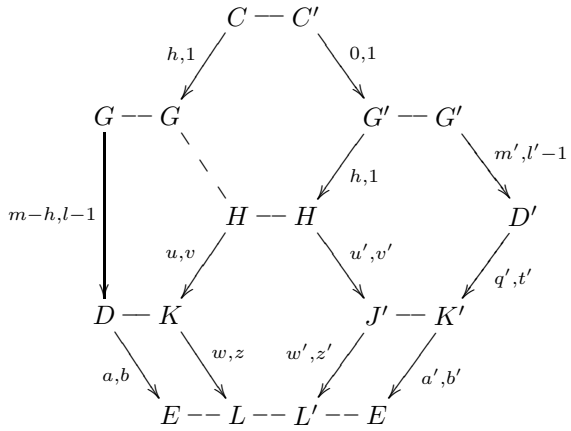
$$m - h + q = u + h' \quad h + u' = m' - h' + q' \quad w + u = w' + u'$$

from which

$$\begin{array}{rcl} q + a & \leq & h' + w \leq h' + u' \leq h' + m' - h' = m' \\ b + t & \leq & z + 1 \leq v' + 1 \leq l' - 1 + 1 = l' \\ q' + a' & \leq & h + w' \leq h + u \leq h + m - h = m \\ b' + t' & \leq & z' + 1 \leq v + 1 \leq l - 1 + 1 = l \\ q + a + m & = & h + h' + u + w = h + h' + u' + w' = m' + a' + q' \end{array}$$

So we can just put $n = a$, $n' = q' + a'$, $k = t + b$, $k' = t' + b'$.

- $\alpha \in \mathscr{O}$, $\beta \in \mathscr{N}$ and there is H with $G \equiv H$ and $G' \to_\beta H$. By applying several times the induction hypothesis, we end up with the following diagram:



together with the equations:

$$\begin{array}{llll} & q' \leq h & w \leq u' & a = w \\ & t' \leq 1 & z \leq v' & b \leq z \\ u \leq m - h & u' \leq m' - h' & w' \leq u & a' = w' \\ v \leq l - 1 & v' \leq l' - 1 & z' \leq v & b' \leq z' \end{array}$$

and

$$u = m - h \quad h + u' = m' + q' \quad w + u = w' + u'$$

from which

$$\begin{array}{rcl} a & \leq & w \leq u' \leq m' \\ b & \leq & z \leq v' \leq l' - 1 \leq l' \\ q' + a' & \leq & h + w' \leq h + u \leq h + m - h = m \\ b' + t' & \leq & z' + 1 \leq v + 1 \leq l - 1 + 1 = l \\ a + m & = & w + u + h = w' + u' + h = w'm' + q' = a' + m' + q' \end{array}$$

So, we can just put $n = a$, $n' = a' + q'$, $k = b$, $k' = t' + b'$.

- The last case is similar to the previous one.

This concludes the proof. □

As a direct consequence of the previous proposition we have:

Proposition 6. *Let C, D, E be configurations with $C \xrightarrow{m} D$ and $C \xrightarrow{n} E$. Then, there is a configuration F with $D \xrightarrow{p} F$ and $E \xrightarrow{q} F$, where $p \leq n$, $q \leq m$ and $p + m = q + n$.*

Finally, we can prove the main result of this section:

Theorem 2. *A configuration C is strongly normalizing iff C is weakly normalizing.*

Proof. Strong normalization implies weak normalization. Suppose, by way of contradiction, that C is weakly normalizing but not strongly normalizing. This implies there is a configuration D in normal form and an m-sequence from C to D. Since C is not strongly normalizing, there is an infinite sequence $C = C_1, C_2, C_3, \ldots$ with $C_1 \to C_2 \to C_3 \to \ldots$ From this infinite sequence, we can extract an $m+1$-sequence, due to Lemma 7. Applying Proposition 6, we get a configuration F and a 1-sequence from D to F. However, such a 1-sequence cannot exist, because D is normal. □

5 Standardizing Computations

One of the main interesting properties of the Q-calculus is the capability of performing computational steps in the following order:

- First perform classical reductions.
- Secondly, perform reductions that build the underlying quantum register.
- Finally, perform quantum reductions.

We distinguish three particular subsets of $\mathscr{L}$, namely $\mathscr{Q} = \{\mathsf{Uq}, \mathsf{q}.\beta\}$, $n\mathscr{C} = \mathscr{Q} \cup \{\mathsf{new}\}$, and $\mathscr{C} = \mathscr{L} - n\mathscr{C}$. Let $C \to_\alpha C'$ and let M be the relevant redex in C; if $\alpha \in \mathscr{Q}$ the redex M is called *quantum*, if $\alpha \in \mathscr{C}$ the redex M is called *classical*.

Definition 11. A configuration C is called *non classical* if $\alpha \in n\mathscr{C}$ whenever $C \to_\alpha C'$. Let NCL be the set of non classical configurations. A configuration C is called *essentially quantum* if $\alpha \in \mathscr{Q}$ whenever $C \to_\alpha C'$. Let EQT be the set of essentially quantum configurations.

Before claiming the standardization theorem, we need the following definition:

Definition 12. A CNQ computation starting with a configuration C is a computation $\{C_i\}_{i<\varphi}$ such that $C_0 = C$ and:
1. for every $0 < i < \varphi - 1$, if $C_{i-1} \to_{n\mathscr{C}} C_i$ then $C_i \to_{n\mathscr{C}} C_{i+1}$;
2. for every $0 < i < \varphi - 1$, if $C_{i-1} \to_{\mathscr{Q}} C_i$ then $C_i \to_{\mathscr{Q}} C_{i+1}$.

More informally, a CNQ computation is a computation when new reductions are always performed after classical reductions and before quantum reductions.

NCL is closed under new reduction, while EQT is closed under quantum reduction:

Lemma 9. *If* $[\mathcal{Q}, \mathcal{QV}, M] \in \mathsf{NCL}$ *and* $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathsf{new}} [\mathcal{Q}', \mathcal{QV}', M']$ *then* $[\mathcal{Q}', \mathcal{QV}', M'] \in \mathsf{NCL}$.

Proof. Let us denote with $\mathbf{C}[\,]$ a generic context that does not contain classical redexes, and let $\mathtt{new}(c)$ be the reduced redex in M.

The proof proceeds by cases on the structure of M.

there are several case:

1. $M \equiv \mathtt{new}(c)$ and $M' \equiv q$.
   Observe that M' is in normal form and conclude
2. $M \equiv \mathbf{C}[L(\mathtt{new}(c))]$ and $M' \equiv \mathbf{C}[Lq]$ Observe that L cannot be $\lambda x.R$ because $M \in \mathsf{NCL}$ and therefore no classical redexes can be generated.
3. in the following cases it is immediate to observe that the reduction does not generate classical redexes:
   (a) $M \equiv \mathbf{C}[(\mathtt{new}(c))L]$ and $M' \equiv \mathbf{C}[qL]$;
   (b) $M \equiv \mathbf{C}[\lambda!x.(\mathtt{new}(c))]$ and $M' \equiv \mathbf{C}[\lambda!x.q]$.
   (c) $M \equiv \mathbf{C}[\langle N_1, \ldots, N_{k-1}, (\mathtt{new}(c)), N_{k+1} \ldots, N_w\rangle]$
   and
   $M' \equiv \mathbf{C}[\langle N_1, \ldots, N_{k-1}, q, N_{k+1} \ldots, N_w\rangle]$
   (d) $M \equiv \mathbf{C}[\mathtt{p\_new}((\mathtt{new}(c)))]$ and $M' \equiv \mathbf{C}[\mathtt{p\_new}(q)]$
   (e) $M \equiv \mathbf{C}[\mathtt{q\_new}((\mathtt{new}(c)))]$ and $M' \equiv \mathbf{C}[\mathtt{q\_new}(q)]$

□

Lemma 10. *If* $[\mathcal{Q}, \mathcal{QV}, M] \in \mathsf{EQT}$ *and* $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathscr{Q}} [\mathcal{Q}', \mathcal{QV}', M']$ *then* $[\mathcal{Q}', \mathcal{QV}', M'] \in \mathsf{EQT}$.

Proof. Let us denote with $\mathbf{C}[\,]$ a generic context that does not contain classical redexes. Let $P \equiv \lambda\langle x_1, \ldots, x_n\rangle.N\langle r_1, \ldots, r_n\rangle$ and $R \equiv U\langle r_1, \ldots, r_n\rangle$ be two quantum redexes and let $N' \equiv N\{r_1/x_1, \ldots, r_n/x_n\}$

The proof proceeds by case on the shape of the reduced redex in M (and by subcases on the structure of M).

case 1: the reduced redex is P

Cause commutative reductions, it is impossible that M is $\mathbf{C}[PL]$.

Let us examine all the possible cases:

1. $M \equiv P$ and $M' \equiv N'$.
   It is trivial to observe that M' cannot contain classical redexes.

2. $M \equiv \mathbf{C}[LP]$ and $M' \equiv \mathbf{C}[LN'] \equiv \mathbf{C}[(L)N\{r_1/x_1, \ldots, r_n/x_n\}]$:
   The reduction could (hypothetically) create a (new) classical redex in $M' \equiv \mathbf{C}[(L)N']$ iff :
   (a) $L \equiv \lambda!z.L'$ and $N' \equiv !N''$: impossible because in this case P should have the shape $\lambda\langle x_1, \ldots, x_n\rangle.!N'''\langle r_1, \ldots, r_n\rangle$, but this term is not well formed;
   (b) $L \equiv \lambda\langle z_1, \ldots, z_r\rangle.L'$ and $N' \equiv \langle N'_1, \ldots, N'_r\rangle$: in this case M should be: $\mathbf{C}[(\lambda\langle z_1, \ldots, z_r\rangle.L')(\lambda\langle x_1, \ldots, x_n\rangle.N\langle r_1, \ldots, r_n\rangle)]$, but this is impossible because in this case M has a commutative redex;
   (c) $L \equiv \lambda\langle z_1, \ldots, z_r\rangle.L'$ and $N' \equiv (\lambda\pi'.N')N''$: impossible because M should be $\mathbf{C}[(\lambda\langle z_1, \ldots, z_r\rangle.L')(\lambda\langle x_1, \ldots, x_n\rangle.N\langle r_1, \ldots, r_n\rangle)]$ and M should have a commutative redex.
   (d) $N' \equiv (\lambda\pi'.N')N''$:
3. in the following cases:
   (a) $M \equiv \mathbf{C}[\lambda\pi.P]$ and $M' \equiv \mathbf{C}[\lambda\pi.N']$.
   (b) $M \equiv \mathbf{C}[\lambda!x.P]$ and $M' \equiv \mathbf{C}[\lambda!x.N']$.
   (c) $M \equiv \mathbf{C}[\langle N_1, \ldots, N_{k-1}, P, N_{k+1} \ldots, N_w\rangle]$
   and
   $M' \equiv \mathbf{C}[\langle N_1, \ldots, N_{k-1}, N', N_{k+1} \ldots, N_w\rangle]$.
   by means of (1) it is immediate to observe that the reduction does not generate classical redex in M'.
4. $M \equiv \mathbf{C}[\mathtt{p\_new}(P)]$ and $M' \equiv \mathbf{C}[\mathtt{p\_new}(N')]$
   It is immediate to observe that the reduction does not generate new classical redexes, in fact by linearity, N' cannot be a neither 0 nor 1.
5. $M \equiv \mathbf{C}[\mathtt{q\_new}(P)]$ and $M' \equiv \mathbf{C}[\mathtt{q\_new}(N')]$
   as for the previous case.

case 2: the reduced redex is R
there are several case:

1. $M \equiv R$ and $M' \equiv \langle r_1, \ldots, r_n\rangle$.
   M' is in normal form.
2. $M \equiv \mathbf{C}[L(U\langle r_1, \ldots, r_n\rangle)]$ and $M' \equiv \mathbf{C}[L\langle r_1, \ldots, r_n\rangle]$
   Observe that L cannot be $\lambda x.R$ because $M \in \mathsf{EQT}$ and therefore no classical redexes can be generated.
3. in the following cases it is immediate to observe that the reduction does not generate classical redexes:
   (a) $M \equiv \mathbf{C}[(U\langle r_1, \ldots, r_n\rangle)L]$ and $M' \equiv \mathbf{C}[\langle r_1, \ldots, r_n\rangle L]$;
   (b) $M \equiv \mathbf{C}[\lambda!x.(U\langle r_1, \ldots, r_n\rangle)]$ and $M' \equiv \mathbf{C}[\lambda!x.\langle r_1, \ldots, r_n\rangle]$.
   (c) $M \equiv \mathbf{C}[\langle N_1, \ldots, N_{k-1}, (U\langle r_1, \ldots, r_n\rangle), N_{k+1} \ldots, N_w\rangle]$
   and
   $M' \equiv \mathbf{C}[\langle N_1, \ldots, N_{k-1}, \langle r_1, \ldots, r_n\rangle, N_{k+1} \ldots, N_w\rangle]$
   (d) $M \equiv \mathbf{C}[\mathtt{p\_new}((U\langle r_1, \ldots, r_n\rangle))]$ and $M' \equiv \mathbf{C}[\mathtt{p\_new}(\langle r_1, \ldots, r_n\rangle)]$
   (e) $M \equiv \mathbf{C}[\mathtt{q\_new}((U\langle r_1, \ldots, r_n\rangle))]$ and $M' \equiv \mathbf{C}[\mathtt{q\_new}(\langle r_1, \ldots, r_n\rangle)]$

□

This way we are able to state the Standardization Theorem.

Theorem 3 (Standardization)**.** *For every computation $\{C_i\}_{i<\varphi}$ such that $\varphi \in \mathbb{N}$ there is a* CNQ *computation $\{C'_i\}_{i<\xi}$ such that $C_0 = C'_0$ and $C_{\varphi-1} = C'_{\xi-1}$.*

Proof. We will build a CNQ computation in three steps:
1. Let us start to reduce $C'_0 = C_0$ *by using* $\mathscr{C}$ reductions as much as possible. By Theorem 2 we must obtain a finite reduction sequence $C'_0 \to_{\mathscr{C}} \ldots \to_{\mathscr{C}} C'_k$ s.t. $0 \leq k\langle\varphi$ and no $\mathscr{C}$ reductions are applicable to C'_k

2. Reduce C'_k *by using* new reductions as much as possible. By Theorem 2 we must obtain a finite reduction sequence $C'_k \to_{\mathsf{new}} \ldots \to_{\mathsf{new}} C'_j$ s.t. $k \leq j < \varphi$ and no new reductions are applicable to C'_j. Note that by Lemma 9 such reduction steps cannot generate classical redexes and in particular no classical redex can appear in C'_j.
3. Reduce C'_j *by using* $\mathscr{Q}$ reductions as much as possible. By Theorem 2 we must obtain a finite reduction sequence $C'_j \to_{\mathscr{Q}} \ldots \to_{\mathscr{Q}} \cdots C'_m$ such that $j \leq m \langle \varphi$ and no $\mathscr{Q}$ reductions are applicable to C'_m. Note that by Lemma 10 such reduction steps cannot generate neither $\mathscr{C}$ redexes nor new redexes and in particular neither $\mathscr{C}$ nor new reductions are applicable to C'_m. Therefore C'_m *is in normal form.*

The reduction sequence $\{C'_i\}_{i<m+1}$ is such that $C'_0 \to_{\mathscr{C}} \ldots \to_{\mathscr{C}} C'_k \to_{\mathsf{new}} \ldots \to_{\mathsf{new}} C'_j \to_{\mathscr{Q}} \ldots \to_{\mathscr{Q}} C'_m$ is a CNQ computation. By Proposition 6 we observe that $C_{\varphi-1} = C'_m$. □

The *intuition* behind a CNQ computation is the following: the first phase of the computation is responsible for the construction of a λ–term (abstractly) representing a quantum circuit and does not touch the underlying quantum register. The second phase builds the quantum register without introducing any superposition. The third phase corresponds to proper quantum computation (unitary operators are applied to the quantum register, possibly introducing superposition). This intuition will become a technical recipe in order to prove a side of the equivalence between Q-calculus and quantum circuit families formalism (see Section 6.2).

6 Expressive Power

In this section we study the expressive power of the Q-calculus, showing that it is equivalent to finitely generated quantum circuit families, and consequently (via the result of Ozawa and Nishimura [12]) we have the equivalence with quantum Turing machines as defined by Bernstain and Vazirani[3]. The fact the considered class of circuit families only contains finitely generated ones is not an accident: if we want to represent an entire family by one single lambda term (which is, by definition, a finite object) we must restrict to families which are generated by a discrete set of gates.

6.1 Encoding Quantum Circuits Families

In this Section we will show that each (finitely generated) quantum circuit family can be captured by a *quantum relevant* term.

6.1.1 Classical Strength of the Q-calculus.

The classical fragment of the Q-calculus has the expressive power of pure, untyped lambda calculus.

Lemma 11. *If* $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathscr{M}} [\mathcal{Q}', \mathcal{QV}', M']$, *then* $\mathcal{Q} = \mathcal{Q}'$ *and* $\mathcal{QV} = \mathcal{QV}'$.

Natural Numbers Natural numbers are encoded as follows:

$$\begin{array}{rcl} \overline{0} & = & \lambda!x.\lambda!y.y \\ \forall n.\overline{n+1} & = & \lambda!x.\lambda!y.x!\overline{n} \end{array}$$

This way, we can compute the successor and the predecessor of a natural number as follows:

$$\begin{array}{rcl} \mathbf{succ} & = & \lambda!z.\lambda!x.\lambda!y.x!z \\ \mathbf{pred} & = & \lambda!z.z!(\lambda!x.x)!\overline{0} \end{array}$$

Indeed:

$$\begin{array}{rcl} \mathbf{succ}\ !\overline{n} & \to_{\mathscr{C}} & \lambda!x.\lambda!y.x!\overline{n} = \overline{n+1}; \\ \mathbf{pred}\ !\overline{0} & \to_{\mathscr{C}} & \overline{0}!(\lambda!x.x)!\overline{0} \to_{\mathscr{C}} \overline{0}; \\ \mathbf{pred}\ !\overline{n+1} & \to_{\mathscr{C}} & \overline{n+1}!(\lambda!x.x)!\overline{0} \to_{\mathscr{C}} (\lambda!x.x)!\overline{n} \\ & \to_{\mathscr{C}} & \overline{n} \end{array}$$

Lists. Given a sequence $M_1, \ldots, M_n$ of terms, we can build a term $[M_1, \ldots, M_n]$ encoding the sequence as follows, by induction on n:

$$\begin{array}{rcl} [] & = & \lambda!x.\lambda!y.y; \\ [M, M_1 \ldots, M_n] & = & \lambda!x.\lambda!y.xM[M_1, \ldots, M_n]. \end{array}$$

This way we can construct and destruct lists in a principled way: terms $\mathbf{cons}$ and $\mathbf{sel}$ can be built as follows:

$$\begin{array}{rcl} \mathbf{cons} & = & \lambda z.\lambda w.\lambda!x.\lambda!y.xzw; \\ \mathbf{sel} & = & \lambda x.\lambda y.\lambda z.xyz. \end{array}$$

They behave as follows on lists:

$$\begin{array}{rcl} \mathbf{cons}\ M[M_1, \ldots, M_n] & \rightarrow_{\mathscr{C}}^* & [M, M_1, \ldots, M_n] \\ \mathbf{sel}\ []!N!L & \rightarrow_{\mathscr{C}}^* & L \\ \mathbf{sel}\ [M, M_1, \ldots, M_n]!N!L & \rightarrow_{\mathscr{C}}^* & NM[M_1, \ldots, M_n] \end{array}$$

By exploiting $\mathbf{cons}$ and $\mathbf{sel}$, we can build more advanced constructors and destructors: for every natural number n there are terms $\mathbf{append}_n$ and $\mathbf{extract}_n$ behaving as follows:

$$\begin{array}{rcl} \mathbf{append}_n[N_1, \ldots, N_m]M_1, \ldots, M_n & \rightarrow_{\mathscr{M}}^* & [M_1, \ldots, M_n, N_1, \ldots, N_m] \\ \forall m \leq n.\mathbf{extract}_n M[N_1, \ldots, N_m] & \rightarrow_{\mathscr{M}}^* & M[]N_m N_{m-1} \ldots N_1 \\ \forall m > n.\mathbf{extract}_n M[N_1, \ldots N_m] & \rightarrow_{\mathscr{M}}^* & M[N_{n+1} \ldots N_m]N_n N_{n-1} \ldots N_1 \end{array}$$

Terms $\mathbf{append}_n$ can be built by induction on n:

$$\begin{array}{rcl} \mathbf{append}_0 & = & \lambda x.x \\ \mathbf{append}_{n+1} & = & \lambda x.\lambda y_1 \ldots.\lambda y_{n+1}.\mathbf{cons}\ y_{n+1}(\mathbf{append}_n x y_1 \ldots y_n) \end{array}$$

Similarly, terms $\mathbf{extract}_n$ can be built inductively:

$$\begin{array}{rcl} \mathbf{extract}_0 & = & \lambda x.\lambda y.xy \\ \mathbf{extract}_{n+1} & = & \lambda x.\lambda y.(\mathbf{sel}y!(\lambda z.\lambda w.\lambda v.\mathbf{extract}_n vwz)!(\lambda z.z[]))x \end{array}$$

Indeed:

$$\begin{array}{rcl} \mathbf{extract}_0 M[N_1, \ldots N_m] & \rightarrow_{\mathscr{M}}^* & M[N_1, \ldots, N_m] \\ \mathbf{extract}_{n+1} M[] & \rightarrow_{\mathscr{M}}^* & M[] \\ \forall m \leq n.\mathbf{extract}_{n+1} M[N, N_1 \ldots N_m] & \rightarrow_{\mathscr{M}}^* & \mathbf{extract}_n M[N_1, \ldots, N_m]N \\ & \rightarrow_{\mathscr{M}}^* & M[]N_m \ldots N_1 N \\ \forall m > n.\mathbf{extract}_{n+1} M[N, N_1 \ldots N_m] & \rightarrow_{\mathscr{M}}^* & \mathbf{extract}_n M[N_1, \ldots, N_m]N \\ & \rightarrow_{\mathscr{M}}^* & M[N_{n+1} \ldots N_m]N_n \ldots N_1 N \end{array}$$

Recursion and Iteration. We now need a term for iteration: $\mathbf{rec}$ is defined as $\mathbf{recaux}!\mathbf{recaux}$, where

$$\mathbf{recaux} = \lambda!x.\lambda!y.y!((x!x)!y).$$

For each term M,

$$\begin{array}{rcl} \mathbf{rec}!M & = & (\mathbf{recaux}!\mathbf{recaux})!M \rightarrow_{\mathscr{M}} (\lambda!y.y!((\mathbf{recaux}!\mathbf{recaux})!y))!M \\ & \rightarrow_{\mathscr{M}} & M!((\mathbf{recaux}!\mathbf{recaux})!M)) = M!(\mathbf{rec}!M) \end{array}$$

This will help us in encodings algorithms via recursion. If one wants to iterate a given function over natural numbers, there is $\mathbf{iternat} = \mathbf{rec}!\mathbf{iternataux}$, where

$$\mathbf{iternataux} = \lambda!x.\lambda!y.\lambda!w.\lambda!z.y!(\lambda!v.w!(x!v!w!z)!v)!z$$

Indeed:

$$\begin{array}{rcl}
\mathbf{iternat}\ \overline{0}!M!N & \rightarrow^*_{\mathscr{M}} & \mathbf{iternataux}!(\mathbf{iternat})\overline{0}!M!N \\
& \rightarrow^*_{\mathscr{M}} & \overline{0}!(\lambda!v.M(\mathbf{iternat}v!M!N)!v)!N \\
& \rightarrow^*_{\mathscr{M}} & N \\
\mathbf{iternat}\ \overline{n+1}!M!N & \rightarrow^*_{\mathscr{M}} & \overline{n+1}!(\lambda!v.M!(\mathbf{iternat}!v!M!N)!v)!N \\
& \rightarrow^*_{\mathscr{M}} & (\lambda!v.M(\mathbf{iternat}!v!M!N)!v)!\overline{n} \\
& \rightarrow^*_{\mathscr{M}} & M(\mathbf{iternat}\ !\overline{n}!M!N)!\overline{n}
\end{array}$$

Definition 13. A function $f : \mathbb{N}^n \to \mathbb{N}$ is representable iff there is a term M_f such that:

- Whenever $M_f\overline{m_1}\ldots\overline{m_n}$ has a normal form N (with respect to $\rightarrow^*_{\mathscr{C}}$), then $N = \overline{m}$ for some natural number m.
- $M_f\overline{m_1}\ldots\overline{m_n} \rightarrow^*_{\mathscr{C}} \overline{m}$ iff $f(m_1,\ldots,m_n)$ is defined and equal to m.

Proposition 7. *The class of representable functions coincides with the class of partial recursive functions (on natural numbers).*

Iteration is available on lists, too. Let $\mathbf{iterlist} = \mathbf{rec}!\mathbf{iterlistaux}$, where

$$\mathbf{iterlistaux} = \lambda!x.\lambda y.\lambda!w.\lambda!z.y!(\lambda v.\lambda u.w(xu!w!z)v)!z$$

Indeed:

$$\begin{array}{rcl}
\mathbf{iterlist}\ []!M!N & \rightarrow^*_{\mathscr{C}} & \mathbf{iterlistaux}!(\mathbf{iterlist})[]!M!N \\
& \rightarrow^*_{\mathscr{C}} & []!(\lambda v.\lambda u.M(\mathbf{iterlist}\ u!M!N)v)!N \\
& \rightarrow^*_{\mathscr{C}} & N \\
\mathbf{iterlist}\ [L,L_1,\ldots,L_n]!M!N & \rightarrow^*_{\mathscr{C}} & [L,L_1,\ldots,L_n]!(\lambda v.\lambda u.M(\mathbf{iterlist}\ u!M!N)v)!N \\
& \rightarrow^*_{\mathscr{C}} & (\lambda v.\lambda u.M(\mathbf{iterlist}\ u!M!N)v)L[L_1,\ldots,L_n] \\
& \rightarrow^*_{\mathscr{C}} & M(\mathbf{iterlist}[L_1,\ldots,L_n]!M!N)L
\end{array}$$

6.1.2 Quantum Relevant Terms.

Definition 14. Let $\mathscr{S}$ be any subset of $\mathscr{L}$. The expression $[\mathcal{Q},M] \Downarrow_{\mathscr{S}} [\mathcal{Q}',M']$ means that $[\mathcal{Q},M] \rightarrow_{\mathscr{S}} [\mathcal{Q}',M']$ and $[\mathcal{Q}',M']$ is in normal form with respect to the relation $\rightarrow_{\mathscr{S}}$. $[\mathcal{Q},M] \Downarrow [\mathcal{Q}',M']$ stands for $[\mathcal{Q},M] \Downarrow_{\mathscr{L}} [\mathcal{Q}',M']$

Confluence and the equivalence between weakly normalizing and strongly normalizing configurations authorize the following definition:

Definition 15. A term M is called *quantum relevant* (shortly *q–rel*) if it is well formed and for each list $![!c_1,...,!c_n]$ there are a quantum register $\mathcal{Q}$ and a natural number m such that $[1, M![!c_1,...,!c_n]] \Downarrow [\mathcal{Q},[r_1,\ldots,r_m]]$.

In other words, a quantum relevant term is the analogue of a pure λ-term representing a function on natural numbers.

Remark 2. It is immediate to observe that the class of *q–rel* terms in not recursively enumerable.

6.1.3 Circuits.

An *n-qubit gate* (or, simply, a *qubit gate*) is a unitary operator $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$, while a $\mathcal{V}$-qubit gate (where $\mathcal{V}$ is a qvs) is a unitary operator $G : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$. We here work with computable operators only. If $\mathcal{G}$ is a set of qubit gates, a *Λ-circuit* K based on $\mathcal{G}$ is a sequence

$$U_1, r_1^1, \ldots, r_{n_1}^1, \ldots, U_m, r_1^m, \ldots, r_{n_m}^m$$

where, for every $1 \leq i \leq m$:

- U_i is an n_i-qubit gate in $\mathcal{G}$;
- $r^i_1, \ldots, r^i_{n_i}$ are distinct quantum variables in Λ.

The Λ-gate U_K *determined by* a Λ-circuit

$$K = U_1, r^1_1, \ldots, r^1_{n_1}, \ldots, U_m, r^m_1, \ldots, r^m_{n_m}$$

is the unitary operator

$$(U_m)_{\langle\langle r^m_1, \ldots, r^m_{n_m}\rangle\rangle} \circ \ldots \circ (U_1)_{\langle\langle r^1_1, \ldots, r^1_{n_1}\rangle\rangle}.$$

Let $(\mathbf{K}_i)_{i<\omega}$ be an effective enumeration of quantum circuits.

A *family of circuits* generated by $\mathcal{G}$ is a triple (f, g, h) where:

- $f : \mathbb{N} \to \mathbb{N}$ is a computable function;
- $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a computable function such that $0 \leq g(n, m) \leq n + 1$ whenever $1 \leq m \leq f(n)$;
- $h : \mathbb{N} \to \mathbb{N}$ is a computable function such that for every $n \in \mathbb{N}$, $\mathbf{K}_{h(n)}$ is a $\{r_1, \ldots, r_{f(n)}\}$-circuit based on $\mathcal{G}$.

A family of circuits (f, g, h) generated by a finite set $\mathcal{G}$ is said to be *finitely generated*.

6.1.4 The Result.

The n-th elementar permutation of m elements (where $1 \leq n < m$) is the function which maps n to $n+1$, $n+1$ to n and any other elements in the interval $1, \ldots, m$ to itself. A term M *computes the n-th elementary permutation on lists* iff for every list $[N_1, \ldots, N_m]$ with $m > n$, $M[N_1, \ldots, N_m] \to^*_{\mathscr{C}} [N_1, \ldots, N_{n-1}, N_{n+1}, N_n, N_{n+2}, \ldots, N_m]$.

Lemma 12. *There is a term M such that, for every natural number n, $M\overline{n}$ computes the $n+1$-th elementary permutation on lists.*

Proof. M is the term

$$\lambda!x.\mathbf{iternat}!x!(\lambda!y.\lambda!z.\lambda w.\mathbf{extract}_1(\lambda q.\lambda s.\mathbf{append}_1(yq)s)w)!(\lambda y.\mathbf{extract}_2(\lambda z.\lambda w.\lambda q.\mathbf{append}_2 zqw)y)$$

This completes the proof. □

Lemma 13. *There is a term M such that, for every list $[!N_1, \ldots, !N_n]$, $M[!N_1, \ldots, !N_n] \to^*_{\mathscr{C}} \overline{n}$.*

Proof. M is the term

$$\lambda x.\mathbf{iterlist}x!(\lambda y.\lambda!z.\mathbf{succ}y)!(\overline{0})$$

This completes the proof. □

Lemma 14. *There is a term M such that for every list $[!N_1, \ldots, !N_m]$:*

$$\begin{array}{rcl} M!\overline{0}![!N_1, \ldots, !N_m] & \to^*_{\mathscr{C}} & !0 \\ \forall 1 \leq n \leq m. M!\overline{n}![!N_1, \ldots, !N_m] & \to^*_{\mathscr{C}} & !N_n \\ M!\overline{m+1}![!N_1, \ldots, !N_m] & \to^*_{\mathscr{C}} & !1 \end{array}$$

Proof. M is the term

$$\lambda!x.\lambda!y.(\mathbf{iterlist}y!(\lambda z.\lambda!w.\lambda!q.(q!(\lambda!s.\lambda r.(s!L_{\geq 2}!L_{=1})r)!L_{=0})z)!(\lambda!z.z!(\lambda!w.!1)!0))!x$$

where

$$\begin{array}{rcl} L_{=0} & = & \lambda t.t!\overline{0} \\ L_{=1} & = & \lambda t.(\lambda!u.!w)(t!\overline{0}) \\ L_{\geq 2} & = & \lambda!u.\lambda t.t!u \end{array}$$

This completes the proof. □

Theorem 4. *For every finitely generated family of circuits (f, g, h) there is a quantum relevant term M such that for each list $![!c_1, ..., !c_n]$, $[1, M![!c_1, ..., !c_n]] \Downarrow [Q, N]$ (where $N = [r_1, \ldots, r_m]$) iff $m = f(n)$ and $Q = U_{\mathbf{K}_{h(k)}}(|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle)$ (where we assume $c_0 = 0$ and $c_{n+1} = 1$).*

6.2 From Q–calculus to Circuits

We prove here the converse of theorem 4. This way we will complete the proof of the equivalence with quantum circuit families.

Let M be a q–rel term, let $![!c_1, ..., !c_n], ![!d_1, ..., !d_n]$ be two lists of bits (with the same length) and suppose $[1, M![!c_1, ..., !c_n]] \Downarrow_{n\mathscr{Q}} [Q, N]$. By applying exactly the same computation steps that lead from $[1, M![!c_1, ..., !c_n]]$ to $[Q, N]$, we can prove that $[1, M![!d_1, ..., !d_n]] \Downarrow_{n\mathscr{Q}} [Q', N]$, where Q and Q' live in the same Hilbert Space $\mathcal{H}(\mathbf{Q}(N))$. Therefore, by means of Church's Thesis, we obtain the following:

Proposition 8. *For each q–rel M there exist a term N and two total computable functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $[1, M![!c_1, ..., !c_k]] \Downarrow_{n\mathscr{Q}} [|r_1 \mapsto c_{g(k,1)}, \ldots, r_{f(k)} \mapsto c_{g(k,f(k))}\rangle, N]$, where we conventionally set $c_0 = 0$ and $c_{n+1} = 1$.*

Let us consider $[\mathcal{Q}, M] \in \mathsf{EQT}$ and let us suppose that $[\mathcal{Q}, M] \Downarrow_{\mathscr{Q}} [\mathcal{Q}', [r_1, \ldots, r_m]]$. The sequence of reductions in this computation allows to to build in an effective way a unitary transformation U_M such that $\mathcal{Q}' = U_M(\mathcal{Q})$. Summarizing, we have the following:

Proposition 9. *Let $[\mathcal{Q}, M] \in \mathsf{EQT}$ and suppose $[\mathcal{Q}, M] \Downarrow_{\mathscr{Q}} [\mathcal{Q}', M']$. Then there is a circuit K such that $\mathcal{Q}' = U_K(\mathcal{Q})$. Moreover, K is generated by gates appearing in M. Furthermore K is effectively generated from M.*

As a direct consequence of propositions 8 and 9 we obtain the following:

Theorem 5. *For each q–rel M there is a quantum circuit family (f, g, h) such that for each list $![!c_1, ..., !c_n]$, if $[1, M![!c_1, ..., !c_n]] \Downarrow [\mathcal{Q}, [r_1, \ldots, r_m]]$ then $m = f(n)$ and $\mathcal{Q} = U_{\mathbf{K}_{h(n)}}(|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle)$.*

7 On the Measurement Operator

In the Q-calculus it is not possible to classically observe the content of the quantum register. More specifically, the language of terms does not include any measurement operator which, applied to a quantum variable, has the effect of observing the value of the related qubit. This is in contrast with Selinger and Valiron's λ_{sv} (where such a measurement operator is indeed part of the language of terms) and with other calculi for quantum computation like the so-called measurement calculus [4] (where the possibility of observing is even more central).

Extending Q-calculus with a measurement operator $\mathtt{meas}(\cdot)$ (in the style of λ_{sv}) would not be particularly problematic. However, some of the properties we proved here would not be true anymore. In particular:

- The reduction relation would be probabilistic, since observing a qubit can have different outcomes. As a consequence, confluence would not be true anymore.
- The standardization theorem would not hold in the form it has been presented here. In particular, the application of unitary transformations to the underlying quantum register could not always be postponed at the end of a computation.

The main reason why we have focused our attention to a calculus without any explicit measurement operator is that the (extensional) expressive power of the obtained calculus would presumably be the same [11].

8 Conclusion and Further Work

We have studied the Q-calculus, a quantum lambda calculus based on the paradigm "quantum data and classical control". Differently from most of the related literature, which focus on semantical issues, we faced the problem of expressiveness, proving the computational equivalence of our calculus with a suitable class of quantum circuit families (or equivalently, with the Quantum Turing Machines à la Bernstein and Vazirani).

We have also given a standardization theorem, that should help clarifying the interaction between the classical and the quantum world (at least in a λ–calculus setting). Syntactical properties of the calculus, such as subject reduction and confluence, have been studied.

The next step of our research will concern the development of type systems. An interesting question is the following: is it possible to give type systems controlling the (quantum) computational complexity of representable functions?

References

[1] T. Altenkirch and J. Grattage. A functional quantum programming language. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, 2005.

[2] J.-L. Basdevant and J. Dalibard. *Quantum mechanics*. Springer-Verlag, Berlin, 2005. Corrected second printing, With 1 CD-ROM by Manuel Joffre.

[3] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

[4] V. Danos, E. Kashefi, and P. Panangaden. *The Measurement Calculus*. http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0412135, 2004.

[5] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.

[6] R. P. Feynman. Simulating physics with computers. *Internat. J. Theoret. Phys.*, 21(6-7):467–488, 1981/82. Physics of computation, Part II (Dedham, Mass., 1981).

[7] S. C. Kleene. λ-definability and recursiveness. *Duke Math. J.*, 2(2):340–353, 1936.

[8] E. Knill. Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory, 1996.

[9] P. Maymin. Extending the lambda calculus to express randomized and quantu- mized algorithms. Technical Report arXiv:quant-ph/9612052, arXiv, 1996.

[10] P. Maymin. The lambda-q calculus can efficiently simulate quantum computers. Technical Report arXiv:quant-ph/9702057, arXiv, 1997.

[11] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, 2000.

[12] H. Nishimura and M. Ozawa. Computational complexity of uniform quantum circuit families and quantum turing machines. *Theor. Comput. Sci.*, 276(1-2):147–181, 2002.

[13] P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[14] P. Selinger and B. Valiron. A lambda calculus for quantum computation with classical control. *Math. Structures Comput. Sci.*, 16(3):527–552, 2006.

[15] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.

[16] A. K. Simpson. Reduction in a linear lambda-calculus with applications to operational semantics. In *RTA*, pages 219–234, 2005.

[17] A. van Tonder. A lambda calculus for quantum computation. *SIAM J. Comput.*, 33(5):1109–1135 (electronic), 2004.

[18] P. Wadler. A syntax for linear logic. In *Mathematical foundations of programming semantics (New Orleans, LA, 1993)*, volume 802 of *Lecture Notes in Comput. Sci.*, pages 513–529. Springer, Berlin, 1994.

[19] A. Yao. Quantum circuit complexity. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 352–360, Los Alamitos, California, 1993. IEEE Press.